Igor Kotenko
Victor Skormin (Eds.)

LNCS 6258

Computer Network Security

5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security, MMM-ACNS 2010
St. Petersburg, Russia, September 2010, Proceedings

Springer

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 15-07-2010	2. REPORT TYPE Final Report	3. DATES COVERED *(From – To)* 27 January 2009 - 27-Feb-10

4. TITLE AND SUBTITLE Workshop on Scientific Analysis and Policy in Network Security	5a. CONTRACT NUMBER FA8655-09-M-4004
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Dr. Igor Kotenko	5d. PROJECT NUMBER
	5d. TASK NUMBER
	5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) St. Petersburg Institute For Informatics & Automation of the Russian Academy of Sciences 39 14th Liniya St Petersburg 199178 Russia	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD Unit 4515 BOX 14 APO AE 09421	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) SPC 09-4004

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.
Copyrighted materials: Copyright Springer-Verlag Berlin Heidelberg 2010

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report entails the Proceedings of the Fifth International Conference Mathematical Methods, Models, and Architectures for Computer Networks Security (MMM-ACNS-2010), and includes the First International Workshop Scientific Analysis and Policy Support for Cyber Security (SA&PS4CS-2010). The main objectives of the MMM-ACNS-2010 Conference are to discuss state-of-the-art in mathematical methods and models for computer networks and information security to promote a better understanding of recent achievements and trends in the computer network security, as well as making use of recent achievements in the area of advanced information technologies for computer network and information assurance. Security assurance of resident information and computer networks' software is one of the important problems of the modern computer science and information technology. The problem importance is confirmed by ever increasing multiplicity and diversity of threads and vulnerabilities of computer networks, permanently increasing significance and value of information itself, and by potentially devastating consequences of successful attacks on integrity, resource availability and information confidentiality. Unauthorized access to computer network facilities and network resources, especially in global networks participating in real-time control operations, may be truly disastrous.

15. SUBJECT TERMS
EOARD, Network Security, Cyber Operations

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT UNCLAS	b. ABSTRACT UNCLAS	c. THIS PAGE UNCLAS	UL	145	JAMES LAWTON Ph. D.
					19b. TELEPHONE NUMBER *(Include area code)* +44 (0)1895 616187

Lecture Notes in Computer Science 6258

Igor Kotenko Victor Skormin (Eds.)

Computer Network Security

5th International Conference on Mathematical
Methods, Models and Architectures for
Computer Network Security, MMM-ACNS 2010
St. Petersburg, Russia, September 8-10, 2010
Proceedings

Springer

Volume Editors

Igor Kotenko
Institution of the Russian Aeademy of Seienees
St. Petersburg Institute for Informaties and Automation of RAS
39, 14-th Liniya, St. Petersburg, 199178, Russia
E-mail: ivkote@comsee.spb.ru

Vietor Skormin
Binghamton University (SUNYI)
Binghamton, NY 13902, USA
E-mail: vskormin@binghamton.edu

Library of Congress Control Number: 2010931166

CR Subjeet Classifieation (1998): C.2, D.4.6, E.3, K.6.5, K.4, H.4, J.1

LNCS Sublibrary: SL 5 – Computer Communieation Networks and
Teleeommunieations

ISSN 0302-9743
ISBN-10 3-642-14705-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-14705-0 Springer Berlin Heidelberg New York

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

This volume contains papers presented at the 5th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS 2010) held in St. Petersburg, Russia, during September 8-10, 2010. The conference was organized by the Institution of the Russian Academy of Sciences St. Petersburg Institute for Informatics and Automation of RAS (SPIIRAS) in cooperation with Binghamton University (SUNY).

The previous conferences in the series (MMM-ACNS 2001, MMM-ACNS 2003, MMM-ACNS 2005 and MMM-ACNS 2007) organized by SPIIRAS and Binghamton University (SUNY) demonstrated the great interest of the international scientific community in the theoretical and practical aspects of computer network and information security.

MMM-ACNS 2010 provided the next international forum for sharing original research results among specialists in fundamental and applied problems of computer network security. A total of 54 papers from 19 countries related to significant aspects of the theory and applications of computer network and information security were submitted to MMM-ACNS 2010: 16 papers were selected for regular and 6 for short presentations (30% of acceptance for full papers and 40% for all papers).

Six technical sessions were organized, namely: security modeling and covert channels; security policies and formal analysis of security properties; authentication, authorization, access control and public key cryptography; intrusion and malware detection; security of multi-agent systems and software protection; adaptive security, security analysis and virtualization. The MMM-ACNS 2010 program was enriched by papers presented by five distinguished invited speakers: Hervé Debar (Institut Telecom – Telecom SudParis, France), Dieter Gollmann (Technical University of Hamburg-Harburg, Germany), Greg Morrisett (Harvard University, USA), Bart Preneel (Katholieke Universiteit Leuven, Belgium), and Ravi Sandhu (University of Texas at San Antonio, USA).

The success of the conference was assured by the team effort of the sponsors, organizers, reviewers and participants. We would like to acknowledge the contribution of the individual Program Committee members and thank the paper reviewers. Our sincere gratitude goes to the participants of the conference and all authors of the submitted papers. We are grateful to our sponsor, the European Office of Aerospace Research and Development (EOARD) of the US Air Force, the US Office of Naval Research Global (ONRGlobal), and the Russian Foundation for Basic Research, for their generous support.

We wish to express our gratitude to the Springer LNCS team managed by Alfred Hofmann for their help and cooperation.

September 2010

Igor Kotenko
Victor Skormin

Organization

General Chairs

Rafael M. Yusupov	Institution of the Russian Academy of Sciences St. Petersburg Institute for Informatics and Automation of RAS (SPIIRAS), Russia
Robert L. Herklotz	US Air Force Office of Scientific Research, USA

Program Committee Co-chairs

Igor Kotenko	Institution of the Russian Academy of Sciences St. Petersburg Institute for Informatics and Automation of RAS (SPIIRAS), Russia
Victor Skormin	Binghamton University, USA

Program Committee

Mikhail Atallah	Purdue University, USA
Fabrizio Baiardi	University of Pisa, Italy
Cataldo Basile	Politecnico di Torino, Italy
Konstantin Beznosov	University of British Columbia, Canada
Julien Bourgeois	University of Franche-Comte, France
Mariano Ceccato	Fondazione Bruno Kessler, Italy
David Chadwick	University of Kent, UK
Shiu-Kai Chin	Syracuse University, USA
Howard Chivers	Cranfield University, UK
Christian Collberg	University of Arizona, USA
Miguel Correia	University of Lisbon, Portugal
Frédéric Cuppens	TELECOM Bretagne, France
Dipankar Dasgupta	University of Memphis, USA
Hervé Debar	Institut Telecom - Telecom SudParis, France
Changyu Dong	Imperial College London, UK
Dennis Gamayunov	Moscow State University, Russia
Dieter Gollmann	Technical University of Hamburg-Harburg, Germany
Stefanos Gritzalis	University of the Aegean, Greece
Alexander Grusho	Moscow State University, Russia
Amir Herzberg	Bar Ilan University, Israel
Ming-Yuh Huang	Northwest Security Institute, USA
Sushil Jajodia	George Mason University, USA
Angelos Keromytis	Columbia University, USA

Victor Korneev	Federal Enterprise "R&D Institute "Kvant", Russia
Klaus-Peter Kossakowski	Presecure Consulting GmbH, Germany
Igor Kotenko	SPIIRAS, Russia
Pavel Laskov	University of Tuebingen, Germany
Javier Lopez	University of Malaga, Spain
Antonio Maña	University of Malaga, Spain
Fabio Martinelli	CNR/IIT, Italy
Gregorio Martinez	University of Murcia, Spain
Catherine Meadows	Naval Research Laboratory, USA
Ann Miller	University of Missouri – Rolla, USA
Nikolay Moldovyan	SPIIRAS, Russia
Wojciech Molisz	Gdansk University of Technology, Poland
Monika Oit	Cybernetica, Estonia
Vladimir Oleshchuk	University of Agder, Norway
Slobodan Petrovic	Gjøvik University College, Norway
Neeli Prasad	Aalborg University, Denmark
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Roland Rieke	Fraunhofer Institute for Secure Information Technology SIT, Germany
Peter Ryan	University of Luxembourg, Luxembourg
Andrei Sabelfeld	Chalmers University of Technology, Sweden
Igor Saenko	SPIIRAS, Russia
Ravi Sandhu	George Mason University and NSD Security, USA
Victor Skormin	Binghamton University, USA
Michael Smirnov	Fraunhofer-Gesellschaft Institute FOKUS, Germany
Artem Tishkov	SPIIRAS, Russia
Bill Tsoumas	Athens University of Economics and Business, Greece
Shambhu Upadhyaya	Buffalo University, USA
Alfonso Valdes	SRI International, USA
Vijay Varadharajaran	Macquarie University, Australia
Valery Vasenin	Moscow State University, Russia
Paulo Verissimo	University of Lisbon, Portugal
Peter Zegzhda	St. Petersburg Polytechnical University, Russia
Cliff Zou	University of Central Florida, USA

Reviewers

Mikhail Atallah	Purdue University, USA
Fabrizio Baiardi	University of Pisa, Italy
Cataldo Basile	Politecnico di Torino, Italy
Konstantin Beznosov	University of British Columbia, Canada
Julien Bourgeois	University of Franche-Comte, France
Mariano Ceccato	Fondazione Bruno Kessler, Italy
David Chadwick	University of Kent, UK
Shiu-Kai Chin	Syracuse University, USA
Howard Chivers	Cranfield University, UK

Christian Collberg	University of Arizona, USA
Miguel Correia	University of Lisbon, Portugal
Frédéric Cuppens	TELECOM Bretagne, France
Dipankar Dasgupta	University of Memphis, USA
Hervé Debar	Institut Telecom - Telecom SudParis, France
Pierpaolo Degano	University of Pisa, Italy
Changyu Dong	Imperial College London, UK
Dennis Gamayunov	Moscow State University, Russia
Dieter Gollmann	Technical University of Hamburg-Harburg, Germany
Stefanos Gritzalis	University of the Aegean, Greece
Alexander Grusho	Moscow State University, Russia
Amir Herzberg	Bar Ilan University, Israel
Ming-Yuh Huang	Northwest Security Institute, USA
Sushil Jajodia	George Mason University, USA
Karthick Jayaraman	Syracuse University, USA
Angelos Keromytis	Columbia University, USA
Markulf Kohlweiss	Katholieke Universiteit Leuven, Belgium
Victor Korneev	Federal Enterprise "R&D Institute "Kvant", Russia
Klaus-Peter Kossakowski	Presecure Consulting GmbH, Germany
Nicolai Kuntze	Fraunhofer Institute for Secure Information Technology SIT, Germany
Pavel Laskov	University of Tuebingen, Germany
Antonio Maña	University of Malaga, Spain
Fabio Martinelli	CNR/IIT, Italy
Gregorio Martinez	University of Murcia, Spain
John McDermott	Naval Research Laboratory, USA
Catherine Meadows	Naval Research Laboratory, USA
Nikolay Moldovyan	SPIIRAS, Russia
Wojciech Molisz	Gdansk University of Technology, Poland
Monika Oit	Cybernetica, Estonia
Vladimir Oleshchuk	University of Agder, Norway
Slobodan Petrovic	Gjøvik University College, Norway
Neeli Prasad	Aalborg University, Denmark
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Willard Thor Rafnsson	Chalmers University of Technology, Sweden
Roland Rieke	Fraunhofer Institute for Secure Information Technology SIT, Germany
Alejandro Russo	Chalmers University of Technology, Sweden
Peter Ryan	University of Luxembourg, Luxembourg
Andrei Sabelfeld	Chalmers University of Technology, Sweden
Igor Saenko	SPIIRAS, Russia
Ravi Sandhu	George Mason University and NSD Security, USA
Michael Smirnov	Fraunhofer-Gesellschaft Institute FOKUS, Germany
Zaharina Stoynova	Fraunhofer Institute for Secure Information Technology SIT, Germany
Artem Tishkov	SPIIRAS, Russia
Bill Tsoumas	Athens University of Economics and Business, Greece

Shambhu Upadhyaya	University at Buffalo, USA
Alfonso Valdes	SRI International, USA
Valery Vasenin	Moscow State University, Russia
Paulo Verissimo	University of Lisboa, Portugal
Peter Zegzhda	St. Petcrsburg Polytechnical University, Russia
Cliff Zou	University of Central Florida, USA

Table of Contents

Invited Papers

Security Modeling and Covert Channels

Security Policies and Formal Analysis of Security Properties

Authentication, Authorization, Access Control and Public Key Cryptography

Intrusion and Malware Detection

Security of Multi-agent Systems and Software Protection

Adaptive Security, Security Analysis and Virtualization

Service Dependencies in Information Systems Security

Hervé Debar[1], Nizar Kheir[2], Nora Cuppens-Boulahia[2], and Frédéric Cuppens[2]

[1] Laboratoire SAMOVAR UMR 5137, Télécom SudParis,
9 rue Charles Fourier 91011 Evry, France
herve.debar@telecom-sudparis.eu
[2] Télécom Bretagne, 2 rue de la Chataigneraie,
35512 Cesson Sévigné Cedex, France
{nizar.kheir,nora.cuppens,frederic.cuppens}@telecom-bretagne.eu

Abstract. In the complex world of information services, we are realizing that system dependencies upon one another have not only operational implications but also security implications. These security implications are multifold. Beyond allowing an attacker to propagate over an information system by leveraging stepping stones vulnerabilities, it also allows a defender to select the most interesting enforcement points for its policies, overall reducing the cost of managing the security of these complex systems. In this paper, we present a dependency model that has been designed for the purpose of providing security operators with a quantitative decision support system for deploying and managing security policies.

1 Introduction

Today's world of information services is becoming more and more reliant on a web of interconnected "unitary" services, whose composition forms the basis of so-called Web Services. In fact, this notion of service composition can be extended to any information system, where the simplest form of application (e.g. a word processor) relies on an operating system, itself relying on a set of hardware components to provide the capability to display, modify, store or print documents. The value to users is the documents, not any of the underlying services, and the user often only realizes the value of the word processor software, including all underlying components within this value.

There have been many efforts to model information systems and their services, including dependencies. Modeling is in fact one of the most common tools used in computer science, for example the Turing machine [1] or the Von Neuman computer [2]. These models have been used to understand the properties of the modeled information system and are largely used today, for example in policy-based management in networks [3,4]. Our work fully embraces the definitions of Policy Enforcement Point (PEP) and Policy Decision Point (PDP) as defined in RFC 2748 [4].

Our work target a specific sub-problem of policy-based management. We wish to model and use the dependencies that naturally exist between the components

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 1–20, 2010.

of an information system. The model expressed above of a standalone computer is very detailed, we can construct coarser or finer models, but in any case we will obtain a model where a user enters a particular service and will trigger a set of dependent actions for the realization of such service. Given the wealth of existing models using graph theory, petri nets and other related formalisms, we believe that the existence of these relations expressed as dependencies is well established.

We furthermore assume that these dependencies have security implications. For at least some of them, a change in the security status of one of the components on each side of a dependency will imply a change in the (security) status of the other component. This implication has multiple consequences and uses. In the scope of our work on countermeasures, we are mostly interested in two of them, finding the proper enforcement points for specific security rules (which then support countermeasure deployment), and computing the impact of attacks and countermeasures that propagate over an information system.

2 State of the Art on Service Dependency Models

2.1 Existing Dependency Models

Model-based management has recently emerged as an important tool to manage large information systems and networks, as well as security properties [5]. following this line of work, or even before, several models have been proposed for dependency modeling.

[6] presents an XML based dependency model. This model provides a backend for building a dependency database, without providing a formal specification of service dependencies. [7] defines a dependency algebra for modeling dependency strengths. It separates the *Dependency* relation from the *Use* relation. It states that critical components should only use and not depend on non-critical components. In [8], a UML-based dependency model describes service dependencies in ad hoc systems. It focuses on the dependencies relevant to ad hoc collaborative environments. Moreover, a service dependency classification for system management analysis is provided in [9]. It separates between functional (implementation-independent) and structural dependencies (implementation-dependent).

More closely related to security, an intruder uses the privileges obtained on the target service in order to increase his benefits [10] in service oriented architectures. Intrusions are compared to black stains which spread in the system through the dependencies available due to attack success. An intrusion impact thus propagates through some dependencies to the target service, but not all dependencies. Existing tree or graph-based service dependency models do not represent conditional impact propagations because they do not implement the privileges which may be obtained by successful attackers on target services. Furthermore, this model limits itself to the propagation of the attacker, not of the impact of the attack.

2.2 Cost Propagation and Response in Service Dependency Models

More recently, several researchers have focused on the evaluation of impact propagation rather than attacker propagation. [11] provides a cost-sensitive approach for balancing between intrusion and response costs. A system map holding dependency information is used as a basis for deciding on response strategy. [12] proposes a function which evaluates intrusion response impacts using dependency trees. It allows a cost-sensitive selection of intrusion responses. Another cost-sensitive analysis of intrusion responses is presented in [13]. It uses dependency graphs instead of dependency trees. Service dependencies are also used for fault analysis [14], dependability analysis [15] and many other applications.

As in [16], informing the response process starts with an accurate assessment of intrusion impacts. While attack graphs trace dependencies between elementary steps constituting an exploit plan, each step is only assigned an abstract cost [17,18,19]. Unless relying on expert knowledge, no formal approach to evaluate elementary costs is provided. It has been shown that service dependencies provide a suitable platform for reasoning about intrusion impacts [20,21,22,23].

While we are clearly inscribed in this line of work, conventional service dependency models, by introducing services as black boxes regrouped in tree [23] or graph [20,21,22] based structures, are unable to catch the way intrusion impacts spread in the system. Instead, they provide only means for propagating availability impacts, but not confidentiality nor integrity. We argue that to better assess intrusion impacts, a representation of service dependencies which includes more than the only information about dependency strengths is required. This is the reason why we introduce the notion of privileges in section 3.2.

2.3 Requirements for Dependency Modeling

The existing dependency models such as graph [11,13,12] or class-based [8] models classify service dependencies using static attributes. These are often informally defined, adapted to only specific system implementations, prone to issues related to their expressiveness and the dependency characteristics they model. The decision process needs to know more than just the existence of a certain dependency and its strength. It needs to model full chains of relationships, taking into account how, when and why a dependency is activated. It must also support the attribution of security properties to these dependencies, to ensure that the different effects of attacks are properly modeled and propagated.

On the other hand, the proposed modeling framework must enable the regrouping of elementary services into dependency blocks with well-defined interfaces. Those blocks can be implemented in other dependency blocks, and thus providing reusability of the dependency model. It must also allow the abstraction of certain dependencies, and thus representing only the dependencies relevant for the application purposes. This capacity for multi-level modeling is imperative to ensure that the proposed models remain actionable, i.e. that they can be understood that the operators that will need to put in practice the results obtained. Our objective being decision support for security operators, we must ensure that modeling is a mean to this end and not an end by itself.

This paper provides a formal representation of service dependencies. It enables the inline evaluation of intrusion costs using both the privileges realized on the target service and its dependencies. The notion of privilege enables the distinction of availability impacts from those of confidentiality and integrity. In the former, the attacker revokes some privileges to legitimate users (e.g., a DoS attack prevents user from accessing to the denied server). In the latter, the attacker seeks to acquire illicit privileges, and thus to have fraudulent access to some assets.

3 Formal Dependency Model

3.1 Simple Service Model

An information service S is generally delivered to users through a protocol specification describing the network interactions, the syntax and the semantics of the messages delivered to the parties (users and service components). Examples of such public specifications are the IETF requests for comments (RFC) (e.g. HTTP, DNS, ...) or web services specifications. A model M_S of a service S is an abstraction describing S using a specific formalism (in our case the AADL formalism [24], see Section 4). The model aims at describing the architecture and behavior of the service while remaining easy to handle for multiple purposes: simulation, proof of properties, management, etc. This informal description is adaptable to many formalisms such as graphs, petri nets, UML diagrams, and many others.

We consider that a service is a somewhat large entity that is build of smaller objects, also called indifferently components (for example software components) or assets (for example information assets of some value). Thus, a model is composed of a set of connected components C_i, some of these components being basic building blocs and others being models by themselves. In a graph representation, the components C_i would be modeled by nodes, and the connections between these components by edges. We also include users in our set of interacting components, in order to obtain a complete model of the service. We thus define the set of components C_S used in the provision of service S as :

$$C_S = \{C_i, i \in \{1 \ldots n\}\} \tag{1}$$

This equation only represents the set of nodes of the graph, not the edges. To introduce the edges, we define the *require* relationship $\Longrightarrow$ between two components C_x and C_y expressed as $C_x \Longrightarrow C_y$, when C_x needs information from C_y in order to deliver its service. This require relationship represents the dependency of C_x over C_y. As specified in section 4.2, C_x is the antecedent component and C_y is the dependent component.

Thus, each component C_i is associated with two sets, the set PC_i of components that provide resources in order to enable C_i to deliver its service to users, and the set SC_i of components that rely on C_i to deliver their service to users. The components in PC_i are the providers of resources to C_i and the components of SC_i are the subscribers of resources provided by component C_i. This relationship is formalized as equations 2 and 3.

$$PC_i = \{C_j \in C_S, C_i \Longrightarrow C_j\} \tag{2}$$

$$SC_i = \{C_k \in C_S, C_k \Longrightarrow C_i\} \tag{3}$$

and our model M_S is finally defined as the set of triples, as shown in equation 4 :

$$M_S = \{(C_i, PC_i, SC_i), i \in \{1 \dots n\}\} \tag{4}$$

While there is redundancy in the specification of the model (components appear in symmetric roles, as apparent in section 4.2), this is equivalent to the specification of the connected input and output interfaces for each component C_i. Each component of PC_i offers an input interface that is connected to the appropriate output interface of C_i. Similarly, C_i offers input interfaces to all components of SC_i. Note that the model developed in section 4 also allows the definition of interfaces that are not connected to any other component. This enables to view basic models as building blocs that can be reused to construct more complete service models. Thus, the proposed formalism is hierarchical. A component can, at a lower level of granularity, be a model itself, as long as the interface used to communicate with it is unique for all the other components.

3.2 The Privileges Extension

In the previous model definition, we simply connect components together in a graph-like fashion. We now extend this definition by adding the following specifications: (1) the privileges granted to the service (consequently the assets it uses), (2) the credentials accredited to this service and (3) the trust it has regarding other privileges and/or credentials. A component is thus defined as $C = (\mathcal{P}r_C, \mathcal{T}r_C, \mathcal{C}r_C)$ where $\mathcal{P}r_C$, $\mathcal{T}r_C$ and $\mathcal{C}r_C$ respectively represent the privileges, trust relationships and credentials implemented by the service C. These implementations specify the access permissions granted to a component and configure the way it interacts with other components through component dependencies. In the remaining of this section, we define the notions of privilege, credential and trust. Further we use these definitions to propose a new representation of component dependencies and the system model as a whole.

We first define an authorization as a *logical right* that applies to some assets. An authorization may be granted to a subject, and thus we introduce the notion of privilege to model the *grant* of a permission to a subject. A privilege is specified by the following rule:

```
Permission(Subj,Act,Obj):-
       Privilege(Priv),Subject(Priv,Subj),
       Authorization(Priv,Auth),Action(Auth,Act),Object(Auth,Obj).
```

A privilege specifies a subject and an appropriate authorization. The latter includes an action which applies to an object. We represent a privilege *Priv* detained by a subject *Subj* with the notation *Subj.Priv*. It is interpreted as:

$$Subj.Priv \Leftrightarrow Privilege(Priv), Subject(Priv, Subj) \tag{5}$$

We use privileges in order to define security objectives in terms of confidentiality (Co), integrity (Ig) and availability (Av). We argue that the assignment of CIA (Confidentiality, Integrity, Availability) cost vectors to critical assets, as in [25], does not provide enough expressiveness. As discussed in [21], Av is not managed the same way as for Co and Ig. Co and Ig are only related to the asset to which they apply, but Av is related to both the asset and the entity which seeks access to this asset.

We specify the security objectives in terms of Co and Ig as cost metrics assigned explicitly to the appropriate assets. They are defined as square cost vectors (Co_i, Ig_i) which apply to the component C_i. The metric Co_i (resp. Ig_i) takes a higher value as the compromise of the Co (resp. Ig) of C_i provokes higher losses to the system.

The resulting cost for illicitly acquiring an authorization α which applies to C_i is evaluated to $max(\mathcal{C}_\alpha \times Co_i, \mathcal{I}_\alpha \times Ig_i)$. $\mathcal{C}_\alpha$ (resp. $\mathcal{I}_\alpha$) is set to null when the authorization α does not disclose (resp. alter) the Co (resp. Ig) of the component C_i.

We specify security objectives in terms of Av by assigning cost scalars to privileges rather than objects. A privilege *S.Priv* is thus critical if the unavailability of the privilege *Priv* to the component *C* (i.e. user) provokes higher losses to the system. While Co and Ig impacts are evaluated according to the authorizations illicitly acquired by an attacker, Av impacts are evaluated according to the privileges which are revoked to their appropriate users. We thus dispose of more granularity to evaluate Av costs because some privileges may be denied to certain, but not all, users.

3.3 Privilege Sharing: Credential and Trust

A credential is an 'entitlement to privilege', it is not coupled to an object, but to an entity which trusts this credential and shares in counterpart some privileges. A credential thus enables an entity which is not assigned some privilege, to share this privilege with some other entities. We introduce credentials with the predicate *Credential* which is defined by the expression:

$$Credential(Cr) \Leftrightarrow \exists(Subj_1, Subj_2) : Owner(Cr, Subj_1), Authority(Cr, Subj_2) \tag{6}$$

In other terms, the credential *Cr* is granted to the subject $Subj_1$, and trusted by the subject $Subj_2$. We represent a credential *Cr* owned by a subject *Subj* with the notation *Subj.Cr*. It is interpreted as:

$$Subj.Cr \Leftrightarrow Credential(Cr), Owner(Cr, Subj) \tag{7}$$

We define trust as an association of a privilege to be shared in counterpart to some credentials and/or privileges. Trust relationships are implemented as part of an authorization scheme by which we may specify the way privileges may be shared between the different subjects of a system. We introduce these relationships using the predicate *Trust* which is defined by the following specification.

```
Trust(Tr) ⇔∃(Subj,Inp,Out):
 Subject(Tr,Subj),Grantee(Tr,Out),Privilege(Out),
 Trustee(Tr,Inp),Credential(Inp)∨Privilege(Inp).
```

In other terms, the subject *Subj* implements a trust relationship by which it shares the privilege *Out* in counterpart to the trusted credential or privilege *Inp*. The satisfaction of a trust relationship results in additional authorizations granted to the trusted subject (i.e. the subject which has the trusted credentials or privileges). The satisfaction of a trust relationship is formalized by the following specification:

```
Subj2.Out:- Trust(Tr),Subject(Tr,Subj),Trustee(Tr,Inp),Grantee(Tr,Out),
   Subject(Out,Subj),[Privilege(Inp),Subject(Inp,Subj2)]∨
   [Credential(Inp),Owner(Inp,Subj2),Authority(Inp,Subj)]
```

Trust relationships are used to configure and set access control associated with service dependencies. The satisfaction of a dependency is constrained by the implementation of appropriate trust relations. These are threatened by attackers who try to bypass those relations.

4 Model Formalism Using AADL

4.1 Introduction to AADL

AADL has been standardized and released by the Society of Automotive Engineers. AADL provides formal modeling concepts for the description and analysis of application system architectures in terms of distinct components and their interactions. We privileged AADL over common modeling languages like UML because AADL provides more powerful features for modeling system runtime behaviors. AADL provides standardized textual and graphical notations for modeling systems and their functional interfaces. It has been designed to be extensible so that analyses that the core language does not support can be supplied. The extensibility in AADL is provided through the *Annex* extension construct.

Our AADL framework models user runtime behaviors when accessing the data provided by dependent services. It contrasts with most functional dependency models since it focuses on the data flows associated with the access to a dependent service rather than on the model of its functional dependencies. This is a key concept in our approach since policy-driven responses require policy enforcement points to deny some of these data flows.

Since our approach focuses on information systems security, we generally avoid to model functional dependencies if these dependencies do not provide a way to alter or enforce security properties. We for example rarely include a dependency between software and the underlying hardware platform, and will also ignore the operating system if it is not part of the managed environment (e.g. in a cloud computing environment). We also rarely model network link properties, unless filtering devices partition the network in zones with different traffic policies.

We thus model services as abstractions, and these are decoupled from the concrete components which realize them. Our decision can be best motivated by the fact that concrete components only introduce functional dependencies which are not relevant in our approach. For instance, a web service is defined through its dependencies, independently whether it is implemented by apache2 server or windows web server. We use for this purpose AADL system abstractions.

AADL models dependencies using inter-component connections. AADL connections reproduce the service topology. They allow modeling multiple service paths through the use of multiple connection paths to the same data. We also use AADL operational modes in order to represent the dependency sequencing during the workflow of the dependent service.

We use the *AADL Error Model Annex* [26] which has also been standardized to add features for modeling the system behavior in the presence of faults. We use faults as model constructs in order to represent the behavior of a dependent service when it can not access to the antecedent service due to a response application. In the remaining of this section, we describe the main elements of our AADL dependency model.

4.2 Specification of Dependencies in AADL

We define a service as the implementation of an interface which *provides* data access to its users (e.g. Web service, IP service). A service often *requires* access to subsidiary data during its normal behavior. It is thus identified through the specification of its required and provided data accesses. We model an elementary service in AADL as a black box with specific *requires/provides* interfaces. Each interface enables a specific data access, either required or provided by the service (see Figure 1). We may add constraints between data required and provided by a service (e.g. the required account is the owner of the provided data). These are expressed as predicates assigned, when necessary, to the corresponding interfaces.

Service *A* depends on service *B* when *A* requires data access which is provided by *B*. *A* is the *dependent* service, and *B* is the *antecedent* service. The failure of

```
-- Implementation of elementary
   service --
system Service_Name
features
RF1: requires data access data_Set_r1;
...
RFn: requires data access data_Set_rn;
PF1: provides data access data_Set_p1;
...
PFm: provides data access data_Set_pm;
end Service_Name;
```

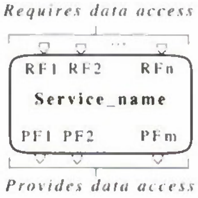


Fig. 1. Elementary Service definition

B, due to an attack or a response, prevents it from providing the data required by A. The proper behavior of A is thus conditioned by the proper behavior of B. Required data accesses enable dependency compliance check: A may never depend on a B if the data access provided by B is not required by A. However, a required data access does not necessarily imply the need for a dependency, because this access can be managed by the service itself. For instance, a mail delivery service requires access to user accounts. These can be managed locally by the service (passwords file), or remotely accessed through a directory service. Only the latter case implies a dependency for the directory service.

We model the dependency of service A to service B by connecting the *provides* interface of B to its complementary *requires* interface of A. The AADL model checks the compliance of this dependency by verifying that the access required by A corresponds to the access provided by B (see Figure 2).

```
system implementation
      Dependency_Model.A
subcomponents
A: system dependent;
B: system antecedent;
connections
const_AB: data access B.PF1 -> A.RF1;
end Dependency_Model.A;
```

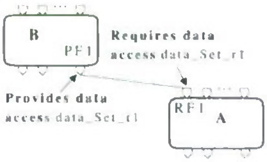


Fig. 2. Explicit Service Dependency Representation

In the formalism of section 3.1, $S = \{A, B\}$ according to equation 1, $PC_A = \{B\}$, $SC_A = \emptyset$, $PC_B = \emptyset$, $SC_B = \{A\}$, and $S = \{(A, \{B'\}, \emptyset), (B, \emptyset, \{A\})\}$ according to equation 4.

5 Dependencies Properties

We define the following dependency characteristics.

Dependency type defines the path of the network flow, and describes the data assets exchanged between the dependent and the antecedent service.

Dependency mode makes precise the occurrence of a dependency within the life cycle and workflow of the dependent service.

Dependency impact evaluates the influence of the absence or degradation of the relation between antecedent and dependent services.

While these characteristics may be completed at a later time, we believe that they are the most relevant for our purpose of using the dependency model for assisting the decision process. In the remainder of this section, we discuss each attribute, and we show how it is modeled in AADL.

5.1 Dependencies Types

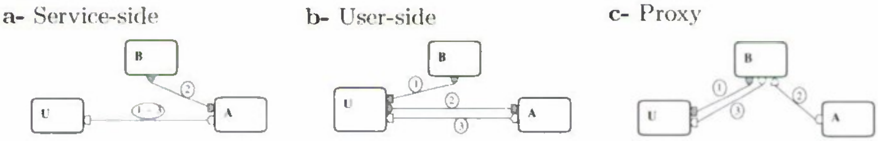
They describe elementary paths followed by the data provided by the antecedent service. They only describe access paths for the direct dependencies of a service. Complete data paths, due to indirect dependencies (dependencies of the direct antecedents of a service), are automatically inferred from elementary access paths for each service.

A dependency type may be either *service-side*, *user-side* or *proxy* dependency.

Service-side dependency: The dependent service initiates the interaction with the antecedent service. The user connects to the dependent service as if no dependency exists (see Figure 3-a).

User-side dependency: The user obtains credentials from the antecedent service and present them to the dependent service. The connection is transparent for the dependent service (see Figure 3-b).

Proxy dependency: The access path to the dependent service is intercepted by the antecedent service. No access path explicitly exists between the dependent service and its user during the dependency (see Figure 3-c).



White interfaces represent the data flow provided by the dependent service for its users.
Gray interfaces represent data flow provided by the antecedent service.
A is the dependent service, *B* is the antecedent service, and *U* is the user of the dependent service.

Fig. 3. Service Dependency Types

5.2 Dependencies Modes

The dependencies mode describes the sequencing of dependencies within the life cycle and workflow of the dependent service. We use AADL operational *modes* for modeling dependency sequencing. AADL *modes* are constructs which represent operational states of a component. Each mode illustrates an operational phase for the dependent service which is characterized by the need for a certain dependency. As such, the dependent service does not notice the failure and/or inaccessibility of the antecedent service unless the former reaches an operational mode where it requires the access to the data provided by the antecedent service. The transition into a dependency mode means that the dependent service has reached an operational phase where it requires access to the data provided by the antecedent service. The transition out of this mode means that the dependency is no longer required.

A service has four operational modes. These modes describe the life cycle of this service. Every dependency mode exists necessarily in at least one of these operational modes. We shall first describe service life cycle in AADL, and later we describe dependency sequencing during this life cycle. The service life cycle holds four operational modes: Start, Idle, Request and Stop modes (see the associated AADL model in Figure 4). They are defined as follows:

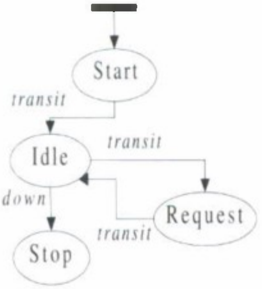
Start Mode characterizes the launching period of a service. The process realizing the service is loading configurations and assets. The transition out of this mode occurs when the process is ready to receive user requests. Dependencies in start mode are one-time dependencies only required during service start-up.

Idle Mode characterizes the period during which a service is waiting for incoming user requests. The transition out of this mode is initiated by a user request, or by a decision to stop the service. The dependencies in this phase are mainly functional dependencies not relevant for the purpose of this paper, but which can be further investigated as for impact evaluations (see section 9).

Request Mode starts when the service receives a user request. It characterizes the in-line dependencies required in order to process this request. The transition from this mode occurs after the user connection is closed.

Stop mode All the actions a service may take before stopping are considered as part of the stop mode.

The time spent in each operational mode varies according to service configurations. Transitions between operational modes may also vary for certain services. For instance, a service configured through the Internet super daemon `inetd` starts on a per-request basis and therefore directly switches to the stop mode at the end of the request mode. The same service started through the boot sequence configuration files `/etc/rc.d/` will run throughout the entire uptime of the system, and will only be in start mode during the boot sequence.

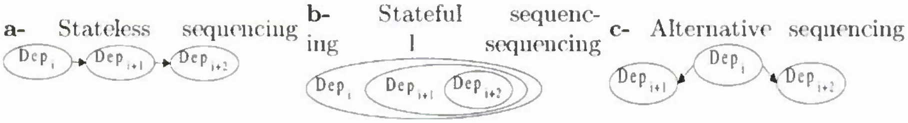


```
system implementation Dependent.instance
 subcomponents
  CStart: system op_State in modes (Start);
  CIdle: system op_State in modes (Idle);
  CRequest: system op_State in modes (Request);
  CStop: system op_State in modes (Stop);
 modes
  Start: initial mode;
  Idle: mode; Request: mode; Stop: mode;
  Start -[CStart.transit]-> Idle;
  Idle -[CIdle.transit]-> Request;
  Request -[CRequest.transit]-> Idle;
  Idle -[CIdle.down]-> Stop;
end Dependent.instance;
```

Fig. 4. Dependent Service Modes

5.3 Dependencies Sequencing

Dependencies in each operational mode are invoked in a certain sequence related to the service behavior. These are defined as AADL operational sub-modes assigned to the components of each operational mode (lines 2-6 in Figure 4). We thus state dependencies within the life cycle of the dependent service, and we determine the dependency sequencing within the same life cycle phase. We obtain a Dependency Finite State Machine (DFSM) with sub-states. Dependencies appear in three possible sequences described as follows.



a- Stateless sequencing

b- Stateful sequencing l sequencing

c- Alternative sequencing

Fig. 5. Service Dependency Sequencing

- *Stateless sequencing*: the satisfaction of the parent dependency is an obligation prior to the access to the child dependency. However, the former does not need to remain satisfied once the latter is accessed (Figure 5-a).
- *Stateful sequencing*: the parent dependency must remain satisfied as long as the child dependency is not satisfied yet (Figure 5-b).
- *Alternative sequencing*: characterizes redundant dependencies. The transition from the parent dependency leads to one of the child dependencies (Figure 5-c).

Stateless and stateful sequencings express conjunctive dependencies. Alternative sequencing expresses disjunctive dependencies where only one alternative dependency is required. Each dependency mode is associated with a specific *require* interface (see Figure 1) which is connected to a specific antecedent service.

5.4 Dependencies Impacts

The dependencies impacts express the consequence of any degradation of the antecedent service, which alters the access to data required by the dependent service. The failure of a dependency alters the transitions between operational modes. This alteration is motivated by the fact that the failure of a dependency denies reaching its subsequent dependencies in case of no alternative dependency.

Dependency failure does not only alter the normal transition out of the failed dependency. It may also restrain the service to switch to another operational mode. For instance, a web server may switch to insecure connections when the SSL service does not respond. We use the AADL error model annex to represent the impact of a dependency failure. Each service is attributed at least two AADL error states, which are normal and failure states. The impact of a dependency is expressed by constraining the transition out of a dependency to occur depending on the error state of the antecedent service. This is done by defining *Guard_Transition* properties which use error propagations. Error propagations are AADL constructs which notify the component at the remote end of a connection about the error state of the other component. We use *Error_Free* and *Failed* propagations which notify respectively an error free and a failed dependency states. Each dependency state may dispose of two transitions. The first is the normal transition, constrained by the satisfaction of the dependency. The second transition is optional. It is constrained by the inability to satisfy the dependency.

6 Dependency Model Framework and Implementation

Section 5 has defined the service dependency characteristics managed using our approach. This section describes the steps for building a dependency model using our framework summarized in Figure 6. We use the Open Source AADL Tool Environment (OSATE)[1] which is a set of Eclipse plug-ins. OSATE maintains AADL models as XML-based files, which allows the reusability of the model.

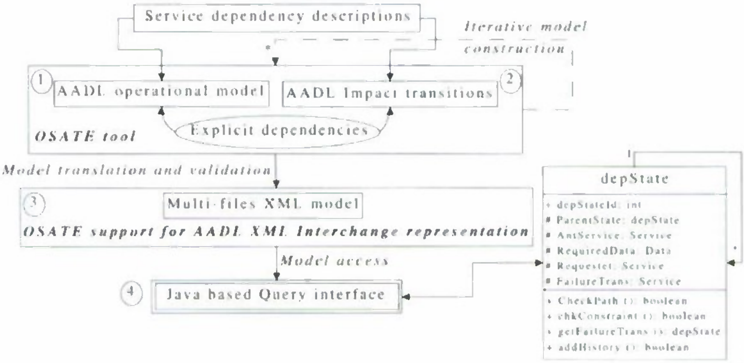


Fig. 6. Dependency Model Framework

The modeling framework is split into four steps. The user is intended to do the first two steps. The last two steps are automatically generated.

Step 1 consists of modeling the explicit dependencies of a service. Each service has a dedicated dependency model defined in an AADL package. Only explicit dependencies are represented. Antecedent services are considered as independent services, and therefore indirect dependencies are not represented.

Step 2 consists of modeling the dependency impacts. Only the impacts of explicit dependencies are modeled. Indirect dependency impacts are inferred from those of explicit dependencies.

[1] `http://la.sei.cmu.edu/aadlinfosite/OpenSourceAADLToolEnvironment.html`

The iteration over the first two steps consists of replacing antecedent services by the implementation of their composite dependency models. Antecedent services, previously used as abstract independent components, are replaced by instantiations of their dependency packages (see the case study for examples).

In **Step 3**, OSATE translates the AADL model into a multi-file XML model. Each package (i.e. elementary dependency model) is saved as an XML file expressed using the AADL XML Interchange format. This step is preceded by an automated model validation. OSATE checks the connections between model components. It flags inappropriate dependencies where a dependent service is made dependent of an antecedent service which does not provide its required data.

Step 4 is the implementation of a query interface which manages the access to the dependency model. This interface is queried for the dependencies of a specific service. We use the Java-based Document Object Model to explore the AADL/XML model. The query interface builds a Dependency Finite State Machine (DFSM) with sub-states in order to represent service dependencies.

The DFSM schema is illustrated in Figure 6. It summarizes all the dependency characteristics modeled in the first two steps. The attributes of a dependency state are (1) the antecedent service, (2) the required data, (3) the requester (dependency type), (4) the dependency impact, (5) the parent dependency and (6) the next dependency (dependency modes). Cyclic dependencies are discarded, and thus a dependency state cannot be a parent for another dependency state which points to the same service.

7 Using Dependencies Models for PEP Selection

In the context of our work on the use of the OrBAC security policy language for intrusion response [27], we have used the proposed model for selecting policy enforcement points as enforcers of OrBAC policy rules.

7.1 Modeling Policy Enforcement Points

The derivation of concrete elementary accesses is followed by a decision process. It aims to reconfigure elementary accesses so that the initial response access rule could be applied. In case of permission, the decision process satisfies at least a minimal set of dependencies. In case of a prohibition, it checks that no dependency path enables the prohibited data access. Access permissions are modified through the reconfiguration of PEPs which are modules associated with services. We therefore consider each service as a PEP having limited access control capabilities. This capability, when it exists, is limited to a specific class of subjects. It thus restrains the PEP capability to apply elementary access rules. For instance, firewall visibility is limited to network level information, it is not able to monitor user-level credentials.

A PEP is able to apply a security rule when (1) the subject in this rule belongs to the capability set of the PEP, (2) the service pointed by the action is managed by the PEP and (3) the object is a data provided by the service. The capability of a PEP depends on its concrete implementation. It is defined as a constraint which must be satisfied by the subject in the security rule. Services which do not have access control capabilities are assigned null capability sets. The PDP may select a certain PEP if the subject within the elementary concrete rule derived for this PEP belongs to its capability class. The PDP selects the optimal response set according to two criteria.

- A prohibition is applied the closer possible to the start state of the DFSM, in order to reduce resource consumption. This is motivated by the fact that when the access is denied at the beginning of the DFSM, subsequent dependency accesses are denied, which contributes in reducing resource consumption.
- The PDP minimizes the configuration changes required for the application of a security rule by minimizing the services which need to be reconfigured.

7.2 Selecting Policy Enforcement Points

S is the set of services obtained from the AADL model. We model the DFSM for the service s_{Dep} as $DFSM_{s_{Dep}} = \{S_a, T_a\}$ where $s_i \in S_a \subset S$ is an antecedent for s_{Dep} and $a_{ij} \in T_a \subset S \times S$ is a transition. A path p_{ij} is a sequence of adjacent transitions which lead from the dependency state s_i to the dependency state s_j. If this path does not exist then $p_{ij} = \phi$. For an input security rule, the PDP crosses $DFSM_{s_{Dep}}$. It searches the minimal set of dependencies which applies the security rule and reduces superfluous resource transactions. Algorithm 1 illustrates the behavior of the PDP. In case of a permission, the PDP searches for the dependency path which requires the least modifications (i.e. reconfigurations) in order to allow the access. The selected path is liberated in order to apply the input permission. In case of a prohibition, the PDP denies all dependency paths. When altering a dependency state, the PDP switches to the failure transition of this state and checks that it does not belong to a permissible path.

8 Using Dependencies Models for Attack Impact Propagation

A service dependency expresses the need for the dependent service to access the antecedent service. The dependent service, which requires some privileges not explicitly assigned to this service (e.g. an online directory service needs access to public data), accesses its antecedent service (e.g. database service) in order to acquire the required privileges (e.g. fetch data).

We formalize the service dependency definition using the *RT* framework in [28], and specifically the RT^D component. RT^D introduces the concept of request which is represented by a delegation credential that delegates from the requester to the request. For example, that *Ea* requests an authorization which belongs to the role *Rb* from *Eb* with its capacity of being empowered in the role *Ra* can be represented by: $Ea \xrightarrow{Ea \ as \ Ra} Eb.Rb$. We use the same delegation concept as

```
input  : $Sr(Type, s, a, o)$
output: $List < s_i, Sr_i > Resp$ with $s_i \in S$
$FSM_a$ = makeTransClosure(getDFSM($a$), $Sr$);
dStart = $FSM_a$.start; dEnd = $FSM_a$.end;
if $Type = Prohibition$ then
    foreach $p_{ij}$ in $FSM_a$ with (i=dStart) & (j=dEnd) do
        if chkRespHistory($p_{ij}$) (returns False if the path has been already intercepted)
        then
            curState = dStart;
            repeat
                curState = curState.getNext($p_{ij}$); returns the next state on the path $p_{ij}$
                if chkCapability($curState$) then
                    Resp.add($curState.AntService$, $curState.Sr$);
                    curState.addHistory($curState.Sr$); add Sr to the resp. history
                    auxPath = $FSM_a$.getPath($curState$.getFailureTrans(), $dEnd$);
                    if $(auxPath \neq \phi) \wedge (curState$.getFailureTrans()$.parent \neq Idle)$ then
                    $p_{ij} \leftarrow$ auxPath;
                end
            until $curState = dEnd$;
        end
    end
else
    In case of permission, the PDP allows the path requiring minimum modifications
    minPath = null; minLength = Infinity;
    foreach $p_{ij}$ in $FSM_a$ with (i=dStart) & (j=dEnd) do
        curLength = 0;
        repeat
            curState = curState.getNext($p_{ij}$);
            if !chkRespHistory($curState$) then curLength ++;
        until $curState = dEnd$;
        if $curLength < minLength$ then {minLength = curLength; minPath = $p_{ij}$;}
    end
    allow($minPath$); Liberates the path in parameter
end
```

Algorithm 1. Evaluation of the resulting impact transfer matrices

in [28], but while replacing roles with privileges. This can be best motivated by the fact that the role concept in role-based management languages is treated as a collection of permissions (i.e. authorizations) [29], which makes it compatible with the privilege concept for service dependencies.

We thus represent a dependency for a service A towards service B by the following specification: $A \xrightarrow{(A.Cr, A.Pr)} B.\mathcal{R}$. It states that the dependent service A, in its faculty of having the credential (Cr) and/or the privilege (Pr), requests the privilege $\mathcal{R}$ from the antecedent service B. We shall note that the dependent service may use more than a single credential and/or privilege to access the antecedent service. These will be specified in the dependency definition. The dependent service, after it satisfies its dependency, acquires additional privileges granted by the antecedent service. The satisfaction of the dependency implies the sharing of the privilege set $\mathcal{R}$ between the dependent and the antecedent services.

We use the definitions of a service dependency and trust in order to specify the condition for a dependency to be satisfied. It is written as:

$$(\mathrm{A} \xrightarrow{A.Cr,A.Pr} \mathrm{B}.\mathcal{R} \Rightarrow \mathrm{A}.\mathcal{R}) \Leftrightarrow \forall\ \mathrm{tr}:(\mathrm{Trust}(\mathrm{tr}),\mathrm{Subject}(\mathrm{tr},\mathrm{B}).\mathrm{Grantee}(\mathrm{tr},\mathcal{R})).$$

$$[(\mathrm{Trustee}(\mathrm{tr},\mathrm{Cr}),\ \mathrm{Owner}(\mathrm{Cr},\mathrm{A}))\vee(\mathrm{Trustee}(\mathrm{tr},\mathrm{Pr}),\ \mathrm{Subject}(\mathrm{Pr},\mathrm{A}))].$$

It states that a dependency is only satisfied when the dependent service uses the credentials and privileges which apply to the trust relationships implemented by the antecedent service.

8.1 Modeling Attacks in the Framework

A privilege is affected either by being illicitly acquired by an attacker or by being denied to its legitimate user. Intrusions are thus introduced in this paragraph as a way by which an attacker alters the privilege assignments. An intrusion either provokes a denial of access to legitimate users and/or provides illegitimate access to the attacker. We define infected privileges as being those which are illegally acquired by the attacker, and revoked privileges as being those which are illegally revoked to the target service, and consequently to all of its users.

We use the vulnerability being exploited within an attack to identify the impact of this attack on the target service. We thus define a vulnerability using the pre/post-condition model, as in [30], by introducing the following attributes: 1. *Target* to represent the vulnerable service, 2. *Access* to represent the vulnerability access vector (i.e. the privileges which must be satisfied by the attacker before he could access the vulnerability), 3. *Infects* to represent privileges for the target service which are infected by the intruder in case the attack succeeds and 4. *Revokes* to represent privileges which are revoked to the target service in case the attack succeeds. We model an attack using the same request statement as for a service dependency. An attacker, with his faculty of having some privileges (i.e. vulnerability access vector), exploits a vulnerability on a target service in order to increase his benefits and thus to acquire additional privileges and/or deny other privileges to the target service. Meanwhile, the success condition of the request is extended in order to include information about the exploited vulnerability. We thus introduce an attack impact using the following specifications:

$$\mathrm{Att} \xrightarrow{Att.Pr} \mathrm{B}.\mathcal{R} \Rightarrow \mathrm{Att}.\mathcal{R} \Leftrightarrow$$

$$\exists \mathrm{v}:\ \mathrm{Vulnerability}(\mathrm{v}),\mathrm{Target}(\mathrm{v},\mathrm{B}).\mathrm{Infects}(\mathrm{v},\mathcal{R}).\mathrm{Access}(\mathrm{v},\mathrm{Pr}),\mathrm{Subject}(\mathrm{Pr},\mathrm{Att}).$$

$$\mathrm{Att} \xrightarrow{Att.Pr} \mathrm{B}.\mathcal{R} \Rightarrow \neg\ (\mathrm{B}.\mathcal{R}) \Leftrightarrow$$

$$\exists \mathrm{v}:\ \mathrm{Vulnerability}(\mathrm{v}).\mathrm{Target}(\mathrm{v},\mathrm{B}).\mathrm{Revokes}(\mathrm{v},\mathcal{R}).\mathrm{Access}(\mathrm{v},\mathrm{Pr}),\mathrm{Subject}(\mathrm{Pr},\mathrm{Att}).$$

We introduce the predicate $Infected(B.\mathcal{R})$ to represent the outcome of the first attack, and the predicate $Revoked(B.\mathcal{R})$ to represent the outcome of the second. We may also explore the correlation of attacks by comparing the outcome of one attack to the access vector of the second, as in [30]. Meanwhile, attack correlation using the privilege model is not among the objectives of this paper. Attack correlation enables the combination of elementary impacts with the level of expertise required for succeeding an attack and the prediction of intrusion objectives in order to foresee additional impacts. This is a subject of interest which must be detailed in a future extension to this study. We are thus interested in this paper in evaluating impacts of elementary (i.e. separated) attacks.

8.2 Attack Impact Propagation

The impact of an attack propagates when components other than the one being attacked are affected by the attack. The attacker acquires (resp. revokes) privileges granted to components other than his target component. He bypasses the trust relations already configured for service dependencies, by using the privileges he already acquired, in order to increase his gain (i.e. system loss). We infer, using the definitions of attacks, dependencies and trust relations, the conditions for attack impact propagation which are summarized in listing 1.

Listing 1. Attack Impact: Propagation of Infections and Revocations

Stmt 1: $\mathrm{Infected}(A.\mathcal{R}) \wedge \exists\ (B,\mathcal{Q}):\ A \xrightarrow{A.\mathcal{R}} B.\mathcal{Q} \Rightarrow \mathrm{Infected}(A.\mathcal{Q})$

Stmt 2: $\mathrm{Revoked}(B.\mathcal{R}) \wedge \exists\ (A,\mathcal{Q}):\ A \xrightarrow{A.\mathcal{Q}} B.\mathcal{R} \Rightarrow \mathrm{Revoked}(A.\mathcal{R})$

Stmt 3: $\mathrm{Revoked}(A.\mathcal{R}) \wedge \exists\ (B,\mathcal{Q}):\ A \xrightarrow{A.\mathcal{R}} B.\mathcal{Q} \Rightarrow \mathrm{Revoked}(A.\mathcal{Q})$

Stmt 4: $\mathrm{Infected}(B.\mathcal{R}) \wedge \exists\ (A,\mathcal{Q}):\ A \xrightarrow{A.\mathcal{Q}} B.\mathcal{R} \Rightarrow \mathrm{Infected}(A.\mathcal{R})$

Statement 1 characterizes an opportunistic attacker who accesses an antecedent service after his attack against a dependent service. The attacker illicitly acquires from the dependent service some credentials and/or privileges which are trusted by the antecedent service. The attacker benefits are thus extended to include all the privileges granted by the antecedent service. Statement 2 illustrates availability propagation. The revocation of some privileges from an antecedent service makes them unavailable for its dependent services. In statement 3, a target service is revoked from some credentials and/or privileges it uses to access an antecedent service. It is thus revoked from the privileges shared by the antecedent service, and so for all the users of the dependent service. Statement 4 characterizes an undisciplined attacker who uses the infected privileges in order to access any dependency and thus to increase his gain.

While impacts iteratively propagate through service dependencies, the resulting attack impact corresponds to all infected ($\forall(u, Pr) : Infected(u.Pr)$) and revoked ($\forall(u, Pr) : Revoked(u.Pr)$) privileges. Our model also evaluates the conjunction of multiple attacks. By separately infecting more privileges, more dependencies could be infected, and so more damages could be inflicted to the system. Since we evaluate also the impact of countermeasures on users, we can compare the impact of attacks and countermeasures candidates to select the best operational compromise.

9 Conclusion

In this paper, we have demonstrated the modeling of dependencies in the context of information systems security, with an application to finding policy enforcement points and to propagating the impact of attacks and countermeasures. While in the paper we limit ourselves to the theoretical aspects of these models, additional work has shown that these models can be of use do model simple services including messaging, authentication and web services.

We are further extending this work towards simulation, in order to compute the impact of attacks and counter-measures on larger information systems. This will enable operators to obtain a decision support tool that iteratively informs them about the costs associated with the current configuration of their information systems, and to help them decide upon configuration changes based on quantitative information.

References

1. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society 2, 230–265 (1936)
2. Goldstine, H.H., von Neumann, J.: On the principles of large scale computing machines. In: Taub, A. (ed.) John von Neumann Collected Works, vol. V, pp. 1–32. The Macmillan Co., New York (1963)
3. Shenker, S., Wroclawski, J.: Network element service specification template. Network Working Group Request for Comments (1997), `http://www.ietf.org/rfc/rfc2216.txt`
4. Boyle, J., Cohen, R., Durham, D., Rajan, R., Herzog, S., Sastry, A.: The cops (common open policy service) protocol. Network Working Group Request for Comments (2000), `http://www.ietf.org/rfc/rfc2748.txt`
5. de Albuquerque, P.J., Krumm, H., de Geus, P.L.: Policy modeling and refinement for network security systems. In: POLICY 2005: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, Washington, DC, USA, pp. 24–33. IEEE Computer Society, Los Alamitos (2005)
6. Ensel, C., Keller, A.: An approach for managing service dependencies with xml and the resource description framework. J. Netw. Syst. Manage. 10, 147–170 (2002)
7. Ding, H., Sha, L.: Dependency algebra: A tool for designing robust real-time systems. In: IEEE International on Real-Time Systems Symposium, pp. 210–220 (2005)
8. Randic, M., Blaskovic, B., Knezevic, P.: Modeling service dependencies in ad hoc collaborative systems. In: Proceedings of EUROCON 2005, pp. 1842–1845. IEEE Computer Society, Los Alamitos (2005)
9. Keller, A., Kar, G.: Dynamic dependencies in application service management. In: Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2000, Las Vegas, NV, USA (2000)
10. Dacier, M., Deswarte, Y., Kaâniche, M.: Quantitative assessment of operational security: Models and tools. In: LAAS Research Report 96493 (1996)
11. Balepin, I., Maltsev, S., Rowe, J., Levitt, K.: Using specification-based intrusion detection for automated response. In: Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection, pp. 136–154 (2003)
12. Toth, T., Kruegel, C.: Evaluating the impact of automated intrusion response mechanisms. In: ACSAC 2002: Proceedings of the 18th Annual Computer Security Applications Conference, Washington, DC, USA, p. 301. IEEE Computer Society, Los Alamitos (2002)

13. Jahnke, M., Thul, C., Martini, P.: Graph based metrics for intrusion response measures in computer networks. In: LCN 2007: Proceedings of the 32nd IEEE Conference on Local Computer Networks, Washington, DC, USA, pp. 1035–1042. IEEE Computer Society, Los Alamitos (2007)
14. Gruschke, B.: Integrated event management: Event correlation using dependency graphs. In: Proceedings of DSOM 1998, Ninth Annual IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Newark, DE, USA (1998)
15. Rugina, A.E., Kanoun, K., Kaâniche, M.: Architecting dependable systems IV. In: A System Dependability Modeling Framework using AADL and GSPNs, pp. 14–38. Springer, Heidelberg (2007)
16. Papadaki, M., Furnell, S.: Informing the decision process in an automated intrusion response system. Information Security Technical Report 10, 150–161 (2005)
17. Noel, S., Jajodia, S.: Managing attack graph complexity through visual hierarchical aggregation. In: CCS Workshop on Visualization and Data Mining for Computer Security (2004)
18. Sheyner, O., Wing, J.: Tools for generating and analyzing attack graphs. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2003. LNCS, vol. 3188, pp. 344–371. Springer, Heidelberg (2004)
19. Noel, S., Jajodia, S., O'Berry, B., Jacobs, M.: Efficient minimum-cost network hardening via exploit dependency graphs. In: Proceedings of the 19th Annual Conference ACSAC (2003)
20. Balepin, I., Maltsev, S., Rowe, J., Levitt, K.: Using specification-based intrusion detection for automated response. In: Proceedings of the 6th International Symposium RAID, pp. 136–154 (2003)
21. Jahnke, M., Thul, C., Martini, P.: Graph based metrics for intrusion response measures in computer networks. In: 32nd IEEE Conference on Local Computer Networks (2007)
22. Kheir, N., Debar, H., Cuppens-Boulahia, N., Cuppens, F., Viinikka, J.: Cost assessment for intrusion response using dependency graphs. In: Proc. IFIP International Conference N2S (2009)
23. Toth, T., Kruegel, C.: Evaluating the impact of automated intrusion response mechanisms. In: Proceedings of the 18th Annual Conference ACSAC (2002)
24. International Society of Automotive Engineers: SAE-AS5506: SAE architecture analysis and design language (2004)
25. Strasburg, C., Stakhanova, N., Basu, S., Wong, J.S.: Intrusion response cost assessment methodology. In: Proceedings of the 4th International Symposium ASIACCS, pp. 388–391 (2009)
26. International Society of Automotive Engineers: SAE-AS5506/1: SAE architecture analysis and design language, error model annex (2006)
27. Thomas, Y., Debar, H., Cuppens, F., Cuppens-Boulahia, N.: Enabling automated threat response through the use of a dynamic security policy. Journal in Computer Virology (JCV) 3, 195–210 (2007)
28. Li, N., Mitchell, J., Winsborough, W.: Design of a role-based trust-management framework. Proceedings of the the 2002 IEEE Symposium on Security and Privacy 1, 114 (2002)
29. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29, 38–47 (1996)
30. Cuppens, F., Autrel, F., Yacine Bouzida, J.G., Gombault, S., Sans, T.: Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework. Annals of Telecommunications 61, 197–217 (2006)

Secure Applications without Secure Infrastructures

Dieter Gollmann

Hamburg University of Technology, Hamburg, Germany
diego@tu-harburg.de

Abstract. The Internet (together with other communications systems) has become a critical infrastructure in industrialized societies. We will examine to which extent this infrastructure needs to be secured for applications to be deployed securely. We will give examples for application layer attacks that cannot be defended against at the infrastructure layer. Hence, deploying a secure infrastructure is not sufficient to protect critical applications. Conversely, we will give examples where an application can be protected without relying on security services provided by the infrastructure. Hence, deploying a secure infrastructure is not necessary to protect critical applications. We will argue that it is only essential for the computing infrastructure to protect its own execution integrity and for the communications infrastructure to offer availability.

Keywords: Critical infrastructures, application security, security engineering.

1 Introduction

It is today common place to observe that industrialized societies have become reliant on IT to such an extent that the Internet (together with other communications systems) has become a critical infrastructure. It would then seem natural that this infrastructure must be protected against attacks; otherwise we could no longer use the services we have become so accustomed to rely on. This view would be supported by the history of IT security, which has important origins in operating systems security and communications security. Both provide protection at the level of IT infrastructures.

We will argue that such a view is mistaken. Protection of the infrastructure is neither necessary nor sufficient to protect applications deployed on the infrastructure. Society relies in the first instance on the services provided by these applications. Hence, critical applications need to be protected. Protection of the infrastructure is only necessary to the extent required by the application. To be precise, we have to start from a risk analysis for a given application and then decide which attacks are best defended against within the application, and when it is better to rely on security services provided by the infrastructure.

We will illustrate this point with a number of case studies. With the advent of the World Wide Web security functions such as access control started to move

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 21–31, 2010.

from the operating system into the browser. This trend is still continuing. Access control in browsers increasingly resembles access control in a traditional operating system. Attacks such as cross-site request forgery and cross-site scripting cause us to move defences from the browser into individual web pages. We will then briefly cover DNS security and in particular DNS rebinding attacks to discuss which security services should be expected from the infrastructure (DNS, in this case) and where the application should protect itself. Finally, we will discuss attacks on applications that rely on SSL/TLS to show that securing communications may not be sufficient for securing the application. In summary, we will make the case that *security is moving to the application layer.*

2 Browser Security

In the 1970s and 1980s work in computer security had a strong focus on operating system security. The operating system can be viewed as an *infrastructure* component providing users and applications with a file system, managing memory, and managing processes. The security services provided by this infrastructure refer in the main to memory and file management. Processes should not be able to read from or write to memory locations allocated to other processes, unless explicitly intended by inter-process communications. Users sharing a machine should get access to files only if permitted by the policy given (multi-user security). Fundamental security concepts such as status information (supervisor/root and user mode), capabilities, and access control lists were developed in this time.

The attacker was a user with (legitimate) access to the operating system interface trying to enhance his privileges or to get illegitimate access to resources. Security features in an application could typically be disabled by an attacker with supervisor permissions at the operating system level, for example by changing the security settings of the application. In this scenario application security intrinsically relies on the security services supplied by the infrastructure.

2.1 Browser Sandbox

This situation changed in the 1990s when the Internet was opened to general use and the first graphical web browsers emerged. The attacker now was a remote entity using the interfaces provided by network protocols and in particular by the web browser. Access control in the *Java sandbox* could constrain code independently of any security services implemented by the operating system. If we treat the browser as an application running on top of an operating system, we have an instance of an application that includes its own protection mechanisms without relying on security services provided by the infrastructure. The *reference monitor* had moved from the operating system into the browser.

2.2 Software Security

At the same time *software security* deficiencies in the operating system started to attract much attention. A remote attacker could, for example, exploit a *buffer* *overrun* vulnerability to run code on a victim's machine [13]. *Ping-of-death* was a denial-of-service attack of the same kind. To defend against such attacks the infrastructure had to be secure in the sense that it could deal with intentionally malformed inputs [9]. In other words, the infrastructure has to guarantee its own *execution integrity* but does not have to supply the application with security services.

3 Web Page Security

A web page is requested by the client's browser through an HTTP request. HTTP cookies included in a request may authenticate the client to the browser. Server-side scripts process request parameters to construct instructions to back-end servers. The response is transmitted from web server to client and rendered by the client's browser. The server may set cookies in a response header. Dynamic web pages contain scripts accepting user input. Scripts may request further server connections. Several attack vectors target this interplay between client and servers.

- An attacker may retrieve cookies from the client, be it to profile the user or to use the cookies to impersonate the client.
- A malicious script in a web page may perform inappropriate operations on the client.
- A malicious script may use the client as a stepping stone to attack a third party.
- A malicious user may send malformed inputs in an HTTP request to perform inappropriate actions with the help of vulnerable server-side scripts (code injection).

3.1 Code Injection Attacks

SQL injection is an example for a code injection attack. A server-side script constructs a SQL query for a back-end database server as a string put together from code fragments that should capture the query logic and from request parameters. Malformed user input in request parameters can change the query logic or insert new database instructions. Note that a single quote terminates strings in SQL. The attacker could thus submit input containing a single quote followed by SQL clauses which would then become part of the query.

To defend against this attack we could either include suitable *sanitization operators* in the script that aim to detect and neutralize malformed inputs. This defence is located firmly within the application. Alternatively, we could modify the infrastructure so that it can protect its own execution integrity. Instead of constructing database queries as strings, queries are precompiled with placeholders for user input. The actual user input is substituted for these placeholders (bound parameters) at runtime.

3.2 Origin Based Access Control

At the client side the browser has become the infrastructure for handling web pages. Today, this infrastructure provides the following security services:

- The browser controls how cookies are included in requests; the widely adopted *same origin policy* states that a cookie may only be included in requests to the domain that had set the cookie.
- The browser controls to which extent a script in a web page may access local memory; in the initial Java sandbox policy a script had no access to local memory; in the Java 2 security model more fine grained access control became possible [8].
- The browser controls where a script in a web page may connect to; again, the *same origin policy* is usually applied to regulate this aspect.

In all three cases the browser performs access control with respect to an origin based security policy. To enforce such a policy, the browser must authenticate the origin of a web page. Current browsers do this in a rudimentary way. They translate between the IP address of the server the page has been received from (more on this in section 4) and the domain name of this server, but there is no fine grained authentication of the individual parts of a web page.

3.3 Cross-Site Scripting and Cross-Site Request Forgery

This shortcoming is exploited by *cross-site scripting* attacks (XSS) [5]. Such an attack uses a 'trusted' server, i.e. a server with more access rights than those granted to the attacker, as a stepping stone. A malicious script might be placed directly in a page on the trusted server (stored XSS, e.g. via a bulletin board). In another version of XSS the script is hidden in a form in a page on the attacker's server. When a victim visits this page a request that contains the hidden script as a query parameter is automatically sent to the trusted server. Should the server mirror this query parameter back to the victim (e.g. in a response to a search) the script is executed in the victim's browser with the access rights of the trusted server (reflected XSS). XSS can be used, for example, to steal cookies from the client.

Authentication of origin has failed as it did not correctly capture the true origin of the attacker's contribution to the page received from the server. Cross-site request forgery attacks targeting a server follow a similar principle [4]. The server has to 'trust' a client, i.e. there has to be an authenticated session (more on this in section 5) where the client has more access rights than those granted to the attacker. The attacker manages to send actions to the server within this session, which are then executed with the access rights of the client.

Client and server could perform authentication at the application layer to defend against this type of attack, rather than relying on the infrastructure provided by browsers and web servers. So-called XSRF prevention tokens are message authentication codes for actions computed from a shared secret that had been established when the session was created. It is essential to store this secret at the client side in a place out of reach for an attacker able to circumvent the browser's origin based security policies. Once more, the application takes care of security and does not rely on a security service provided by the infrastructure.

4 DNS Security

The Domain Name System (DNS) is, in a nutshell, a distributed directory service managing information about so-called *domain names*. Its core service is the mapping from host names to IP addresses, performed for each domain by one of the *authoritative name servers* for that domain. The DNS is a critical infrastructure for the World Wide Web. Users rely on a correct binding from host names to IP address to get access to the services they wish to use. Browsers rely on correct bindings when enforcing origin based security policies.

4.1 Cache Poisoning

There are two types of attacks that break the correct binding between host names and IP addresses. On one side there are the 'traditional' attacks impersonating an authoritative name server to forge IP addresses in the domain of that server. *Cache poisoning attacks* exploit certain features of the DNS, including the caching strategy of resolving name servers and a challenge-response authentication that relies only on the unpredictability of challenges, to achieve this goal. A particularly effective cache poisoning attack using so-called additional resource records is due to Dan Kaminsky[1]. Defences against cache poisoning attacks can be provided at the infrastructure level, e.g. by running separate resolving and authoritative name servers in a domain, by designating random ports for replies from the authoritative name server as to increase unpredictability, and ultimately by having the response from the authoritative name server digitally signed (DNSSec, RFC 4033 to RFC 4035 [1,2,3]).

4.2 DNS Rebinding

There is a second type of attack where an authoritative name server is the source of incorrect bindings. Such *DNS rebinding attacks* and were first discussed in [6]. DNS rebinding attacks exploiting features of browser plug-ins are described in [10]. With DNS rebinding the attacker circumvents origin based policies in the client browser. For example, a script from a page hosted by the attacker may connect to a victim's IP address the browser accepts to be in the attacker's domain because it has been told so by the attackers authoritative name server.

The client browser would have to double check with the host at the designated IP address whether it considers itself to be in the attacker's domain. It must also

[1] For details see e.g.,
`http://unixwiz.net/techtips/iguide-kaminsky-dns- vuln.html`

be noted that it is an intrinsic problem if the client accepts policy information from a third party without checking its veracity.

More generally, we may ask whether it is necessary for an application to rely on the DNS to provide an authenticated binding between a name (not necessarily a domain name) and an IP address. Alternatively, we could split the task of a *rendezvous service* that binds a name to an unauthenticated IP address from the task of an *authentication service* verifying that an address given indeed belongs to that name. The security property expected from the infrastructure would then be availability, which might be achieved by running multiple independent rendezvous services. Address authentication could be implemented in the application layer, e.g. based on secrets shared between client and server such as a user password.

5 Secure Sessions

Besides operating systems security, communications security has been the second main pillar of information security. Protocol suites such as SSL/TLS (TLS v1.2, [7]) or IPsec [11] facilitate the establishment of secure channels between two parties that are connected via an insecure network. More precisely, the threat model assumes an attacker that can read, delete, insert, modify, and replay traffic; direct attacks against end systems are, however, not considered.

In the 1990s distributed applications were 'secured' by running the application over SSL. *https* is a prime example for this pattern: a secure web page is a page accessed via an SSL/TLS channel. Application security builds directly on security services provided by the communications infrastructure.

This approach has two shortcomings. Many end systems are not well secured. This invalidates one major assumption of the threat model that underpins traditional communications security. Arguably, it is more realistic to assume that the communications system is secure but current end systems are not, rather than the other way round. Section 2 has already hinted at this problem. Secondly, attempts at linking concurrent sessions established at different protocol layers may fail.

Consider the following procedure for establishing a mutually authenticated application layer session between a user and a server that share a secret password. First, the user's client establishes an SSL/TLS channel with the server (host). In this step the client's browser checks that the distinguished name in the server certificate matches the host visited and that the certificate is still valid. The user then sends the password via the SSL/TLS channel; the server authenticates the user and returns a HTTP cookie to the client. This cookie is included in future requests issued within the application layer session. The server takes the cookie as evidence that the requests are coming from the user previously authenticated. The EAP-TTLS protocol gives a concrete implementation of this authentication pattern (Figure 1).

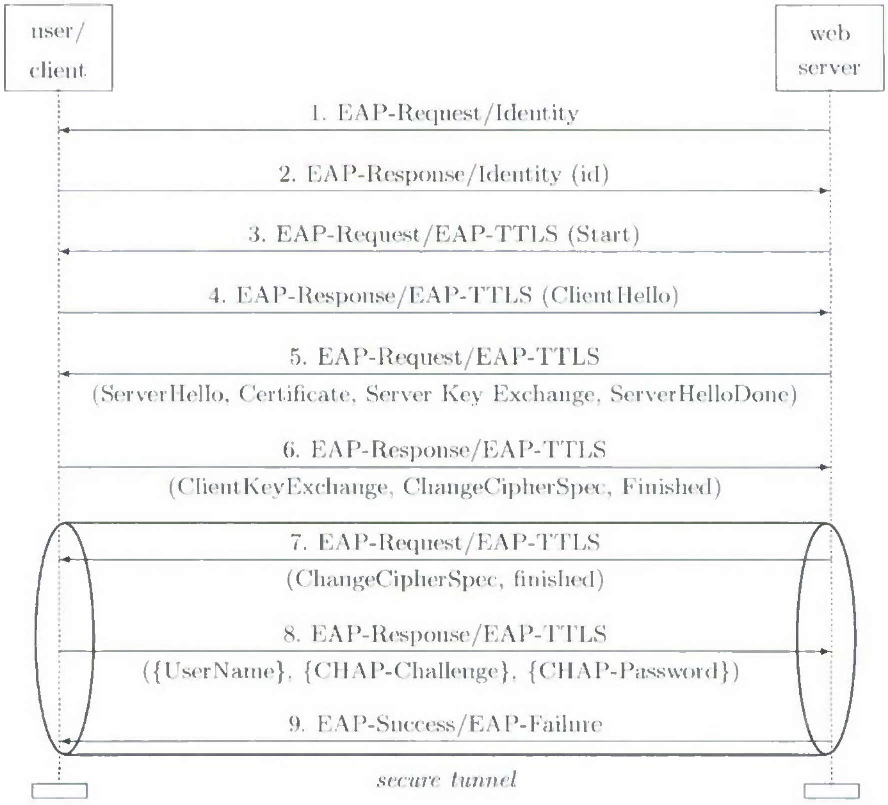

user/
client
web
server
1. EAP-Request/Identity
2. EAP-Response/Identity (id)
3. EAP-Request/EAP-TTLS (Start)
4. EAP-Response/EAP-TTLS (ClientHello)
5. EAP-Request/EAP-TTLS
(ServerHello, Certificate, Server Key Exchange, ServerHelloDone)
6. EAP-Response/EAP-TTLS
(ClientKeyExchange, ChangeCipherSpec, Finished)
7. EAP-Request/EAP-TTLS
(ChangeCipherSpec, finished)
8. EAP-Response/EAP-TTLS
({UserName}, {CHAP-Challenge}, {CHAP-Password})
9. EAP-Success/EAP-Failure
secure tunnel

Fig. 1. EAP Tunneled TLS: EAP-TTLSv0 with CHAP

5.1 Man-in-the-Middle Attacks

A protocol such as EAP-TTLS achieves its goal as long as the SSL/TLS channel has as its endpoint the server holding the password. This is not guaranteed by the protocol itself. Server authentication during the SSL/TLS handshake just guarantees that the server has a valid certificate. It is up to the user to make sure that the host is the one intended.

This check is not always straightforward; host names are not always indicative of service offered. Furthermore, there exist various ways of luring users into connecting to the wrong server. For example, an attack[2] targeting traders with the German Emissions Trading Authority (DEHSt) started from an email purporting to come from a security manager requesting an upgrade to improved security standards[3].

[2] First report on http://www.ftd.de/unternehmen/finanzdienstleister/:gestohlene-co2-zertifikate-hacker-greifen-emissionshaendler-an/50069112.html

[3] This mail – in German – can be found at http://verlorenegeneration.de/2010/02/03/dokumenation-die-phishing-email-im-emissionshandels-hack/

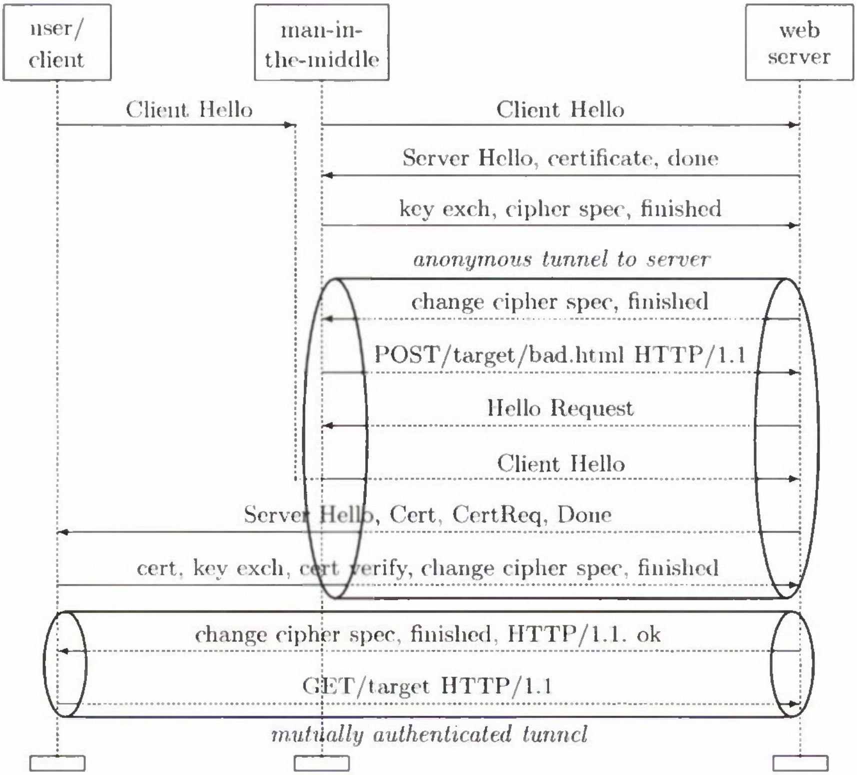

user/ client
man-in- the-middle
web server
Client Hello
Client Hello
Server Hello, certificate, done
key exch, cipher spec, finished
anonymous tunnel to server
change cipher spec, finished
POST/target/bad.html HTTP/1.1
Hello Request
Client Hello
Server Hello, Cert, CertReq, Done
cert, key exch, cert verify, change cipher spec, finished
change cipher spec, finished, HTTP/1.1. ok
GET/target HTTP/1.1
mutually authenticated tunnel

Fig. 2. Man-in-the-middle attack exploiting TLS session renegotiation

Once a user is lured into establishing an SSL/TLS channel with the attacker, the attacker can act as a man-in-the-middle establishing its own SSL/TLS channel with the server. Authentication requests from the server are passed on to the user; the user's response is forwarded to the server; the cookie from the server is sent back to the man-in-the-middle who now can hijack the user's application layer session. A possible countermeasure are cookies tied also to the SSL/TLS channel as proposed in [14]. In the presence of a man-in-the-middle attack client and server use different SSL/TLS channels and could thus detect that cookies are not received in the same channel as they had been originally sent.

A man-in-the-middle attack in time is described in [12]. It exploits a particular usage of SSL/TLS for controlling access to protected resources on a web server. Here, client and server are in possession of certificates. The client initially gets anonymous access to a secure web site by establishing an SSL/TLS channel with server authentication only. When the server receives a request for a protected resource, SSL/TLS session renegotiation is triggered with a *Hello Request* message. In the new session the server asks for client authentication.

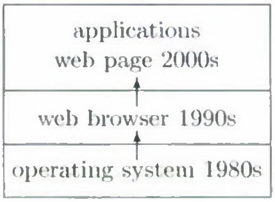


Fig. 3. The reference monitor is moving into the web page

In the man-in-the-middle attack (Figure 2) the attacker waits for a session initiation from the client. The client's message is suppressed and the attacker starts its own session with the server. The attacker sends a request for a protected resource (in Figure 2 a web page is posted to the server) whereupon the server triggers session renegotiation. From this time on the attacker acts as relay between client – that is in the process of establishing a new channel – and server until both have established a new mutually authenticated SSL/TLS channel. A request sent in this new channel will be attributed correctly to the authenticated user and executed with that user's access rights. The attacker's HTTP request had been constructed so that it would be a prefix to the next request in the current session and will now also be executed with that user's access rights.

Note that RFC 5246 does not promise any link between sessions when defining TLS renegotiation. Application designers who had used renegotiation to 'upgrade' the authentication status of the client had thus assumed a service not provided by the infrastructure. To address this situation, RFC 5746 [15] defines a TLS extension where renegotiations are cryptographically tied to the TLS connections they are being performed over. In this case, the infrastructure has followed to meet the – initially unwarranted – expectations of an application.

6 Conclusion

Security is moving to the application layer. Once, the design of secure operating systems and Internet security protocols were the main foundations of information security. According to the then predominant mood, IT systems could be used securely once secure infrastructures were in place. Remnants of this era can still be found in claims that one MUST secure operating systems or the Internet to be able to securely use today's critical IT infrastructures.

We observe, however, that security mechanisms in end systems are moving to the application layer of the software stack. The reference monitor has moved from the operating into web pages (Figure 3). Security components at upper layers may be effective without support from below. This works as long as direct access to the lower layers need not be considered as a threat. In parallel, communications security mechanisms have been moving to the application layer of the protocol stack. At the point where end system security and communications security meet, i.e. in the software components running network protocols, we have seen shared secrets moved up to the application layer to defend against attacks at the infrastructure layer (Section 3.3), contrary to the conventional security strategy that tries to embed secrets at a layer as low as possible, e.g. in tamper resistant hardware.

The security services expected from the infrastructure may thus change over time. We can also observe that our view of what constitutes the infrastructure may change over time. The web browser that started as a new application has today become an essential infrastructure component for Web services.

The Internet is a critical infrastructure because it is the platform for critical applications. Our primary challenge is the protection of these applications. Security services provided by the infrastructure may help in this cause, but trust in these services may also be misplaced when application writers misunderstand the security properties actually guaranteed.

It is a trivial observation on security engineering that defenders ought to know where their systems will be attacked. When attacks are launched via the interface of web applications, the first line of defence should be at that layer.

Attacks may be directed against the application, e.g. fraudulent bank transfers in an e-banking application. Application-level access control necessarily relates to principals meaningful for the application. There may be mappings from those principals to principals known to the infrastructure so that security services from the infrastructure can support application security. However, in every instance we must verify that it is not possible for attackers to redefine the binding between principal names at different system layers. We may thus surmise that it is more likely to find access control solutions at the application layer, as borne out by our earlier observations on reference monitors. In this respect, we have secure applications without a security infrastructure.

Attacks may be directed against the end system hosting the application. Software vulnerabilities in an application may present the attacker with an opportunity to step down into the infrastructure. Although software security issues could be addressed in each application, it would be desirable to have a 'secure' computing infrastructure, i.e. an infrastructure that can deal with malformed inputs forwarded via the applications. In this respect, critical applications benefit from a computing infrastructure that can protect its own execution integrity.

The primary property required from the communications infrastructure is availability. Security services such as confidentiality, integrity, or authenticity may or may not be provided by the communications infrastructure. The relative merits of delivering these services in the various layers of the network stack have been discussed extensively in the research literature. The most relevant issue for critical applications are the choice of relevant principals that can serve as logical endpoints for application layer transactions, and the authentication of those principals.

We leave the reader with a final challenge. When security is moving to the application layer, responsibility for security will increasingly rest with application writers and with end users. At this point in time, neither of the two communities is well prepared to take on this task, but nor has security research made much progress in explaining to non-experts the implications of security decisions.

References

1. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS security introduction and requirements. RFC 4033 (March 2005)
2. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: Protocol modifications for the DNS security extensions. RFC 4035 (March 2005)
3. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: Resource records for the DNS security extensions. RFC 4034 (March 2005)
4. Burns, J.: Cross site reference forgery. Technical report, Information Security Partners, LLC, Version 1.1 (2005)
5. CERT Coordination Center. Malicious HTML tags embedded in client web requests (2000), http://www.cert.org/advisories/CA-2000-02.html
6. Dean, D., Felten, E.W., Wallach, D.S.: Java security: from HotJava to Netscape and beyond. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 190–200 (1996)
7. Dierks, T., Rescorla, E.: The TLS protocol – version 1.2. RFC 5246 (August 2008)
8. Gong, L., Dageforde, M., Ellison, G.W.: Inside Java 2 Platform Security, 2nd edn. Addison-Wesley, Reading (2003)
9. Howard, M., LeBlanc, D.: Writing Secure Code, 2nd edn. Microsoft Press, Redmond (2002)
10. Jackson, C., Barth, A., Bortz, A., Shao, W., Boneh, D.: Protecting browsers from DNS rebinding attacks. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 421–431 (2007)
11. Kent, S., Seo, K.: Security architecture for the Internet protocol, RFC 4301 (December 2005)
12. Marsh, R., Dispensa, S.: Renegotiating TLS. Technical report, PhoneFactor Inc., Malvern (November 2009)
13. One, A.: Smashing the stack for fun and profit. Phrack Magazine 49 (1996)
14. Oppliger, R., Hauser, R., Basin, D.A.: SSL/TLS session-aware user authentication. IEEE Computer 41(3), 59–65 (2008)
15. Rescorla, E., Ray, M., Dispensa, S., Oskov, N.: Transport layer security (TLS) renegotiation indication extension, RFC 5746 (February 2010)

Integrating Types and Specifications for Secure Software Development

Greg Morrisett

Harvard University,
Cambridge, Massachussetts, 02138, USA
greg@eecs.harvard.edu
http://www.eecs.harvard.edu/~greg

Abstract. Today, the majority of security errors in software systems are due to implementation errors, as opposed to flaws in fundamental algorithms (e.g., cryptography). Type-safe languages, such as Java, help rule out a class of these errors, such as code-injection through buffer overruns. But attackers simply shift to implementation flaws above the level of the primitive operations of the language (e.g., SQL-injection attacks). Thus, next-generation languages need type systems that can express and enforce application-specific security policies.

Keywords: dependent types, verification, software-security.

1 Overview

> *In theory, there is no difference between theory and practice. But, in practice, there is.*
>
> Jan L. A. van de Snepscheut

Most security problems today are rooted in implementation errors: failure to check that an array index stays in bounds, failure to check that an input string lacks escape characters, failure to check that an integer passed to an allocation routine is positive, *etc.* Furthermore, the techniques we use for validating that code is free from these errors (fuzz testing, manual inspection, static analysis tools, *etc.*) have proven woefully inadequate. For example, in spite of a large security push starting in 2002, hackers are still finding buffer overruns in Microsoft's operating system and other applications.

The irony is that many of the simplest kinds of errors, such as buffer overruns, could be prevented by the use of a type-safe language instead of C or C++. This is because a type-safe language is required to enforce the basic abstractions of the language through a combination of static and dynamic tests. Languages such as Java, Scheme, and ML are all examples of languages where, at least in principle, buffer overruns cannot occur.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 32–35, 2010.

However, in practice there are four problems with today's type-safe languages:

1. It is expensive to re-write programs in new languages. For example, Windows consists of more than 60 million lines of code.
2. Today's type-safe languages perform poorly when compared to C or C++, particularly for systems-related tasks (e.g., operating systems, networking, databases, etc.)
3. Today's languages check for buffer overruns at run-time and throw an exception that is rarely caught. This shifts the flaw from a possible code injection to a denial of service attack.
4. The type systems for today's languages are too weak to enforce policies needed to stop next-generation attacks.

The first problem is a key issue for legacy systems, but not for next-generation environments (*e.g.*, cell phones, tablets, *etc.*) Furthermore, for key segments of the software market, notably the medical, military and financial industries, the cost of developing highly secure, new software is practical.

The other three problems require fundamental new research in the design of systems programming languages. On the one hand, we need a way to express rich, application-specific security policies and automatically check that the code respects those policies. On the other hand, we need languages that, like C and C++, provide relatively direct access to the underlying machine for performance-critical code.

2 Refinement Types

A number of researchers are looking at next-generation programming languages that support *refinement types*. Refinement types take the form "$\{x : T \mid P(x)\}$" where T is a type and P is a predicate over values of the type T. For example, the type $\{x : \texttt{int} \mid x > 0\}$ captures the set of all positive integers.

The principal challenge with refinement types is finding a way to support type-checking. Some languages, such as PLT Scheme [1], rely upon dynamic checks, so that when a value is "cast" to have a refinement type, the predicate is evaluated on the value, and if it fails, an exception is thrown. This is a simple and expedient way to incorporate refinements, but leads to a number of problems.

First, it restricts the language of predicates that we can use to a decidable fragment. Second, the semantics of these run-time checks are not clear, especially when the predicates can have side effects or when the predicates involve mutable data shared amongst threads. Third, as noted with array-bounds checks, the potential for dynamic failure (an exception) provides for a possible denial of service attack.

To address this last problem, many systems, such as JML# [2] try to discharge these tests at compile time using an SMT theorem prover. In practice, this works well for simple predicates (e.g., linear constraints on integers), but less well for "deep predicates" (e.g., this string is well-formed with respect to this grammar.) Furthermore, SMT provers are focused on fragments of first-order logic (with particular theories). In practice, we have found that, just as programs need to abstract over other sub-programs, specifications need support for abstractions, and higher-order logics provide a powerful way to achieve this.

3 Type-Theoretic Refinement

Proof assistants, such as Coq [3], provide a powerful, uniform way to write (a) programs, (b) specifications and models that capture desired properties of programs ranging from simple typing and safety properties up to full correctness, and (c) formal, machine-checked proofs that a given program meets its specification. They sacrifice automation for finding proofs that code is well-formed, relying instead upon programmers to explicitly construct these proofs. In this sense, they are less convenient than fully automated type-checking techniques. But they are far less limited than the approaches listed above.

For example, Xavier Leroy and his students have used Coq to construct an optimizing compiler that translates a (well-defined) subset of C to PowerPC code, defined operational semantics for both C and PowerPC code, and mechanically proved that when the compiler succeeds in producing target code, that code behaves the same as the source code, thereby establishing the correctness of the compiler [4]. Coq is not alone in providing support for this style of program development: Other examples include ACL2 [5], Agda [6], Epigram [7], and Isabelle [8].

Nevertheless, today's proof assistants suffer from a number of limitations that limit their applicability. One serious shortcoming is that we are limited to writing and reasoning about only purely functional programs with no side effects, including diverging programs, mutable state, exceptions, I/O, concurrency, etc. While some programming tasks, such as a compiler, can be formulated as a pure, terminating function, most cannot. Furthermore, even programs such as a compiler need to use asymptotically efficient algorithms and data structures (*e.g.*, hash-tables) but current dependently typed languages prevent us from doing so. Thus a fundamental challenge is scaling the programming environments of proof assistants to full-fledged programming languages.

4 Ynot

For the past few years, my research group has been investigating a design for a next-generation programming language that builds upon the foundation provided by proof-assistants. We believe that environments, such as Coq, that provide powerful tools for specification and abstraction provide the best basis moving forward, and thus the central issues are (a) how to incorporate support for computational effects, and (b) how to scale proof development and maintenance to real systems.

In the case of effects, we developed a modest extension to Coq called Ynot, which is based on Hoare Type Theory (HTT) [9]. HTT makes a strong distinction between types of pure expressions, and those that may have side-effects, similar to the modality found in Haskell's monadic treatment of IO and state. Impure expressions are *delayed*, and their effects only take place when they are explicitly run. Because impure expressions are delayed, they can be treated as "pure" values, avoiding some of the problems with refinements in the presence of effects.

In addition to extending Coq with support for effects, the Ynot project has investigated techniques for effective systems programming. For example, we built a small, relational database management system using Ynot which was described in previous work [10]. This included an optimizing query compiler, as well as complicated, pointer-based data structures including hash-tables and B+-trees. The whole development, including the parser for queries, the query optimizer, the data structures, and execution engine are verified for partial correctness.

Our experience building verified systems software in this fashion is promising, but a number of hard issues remain to be explored. First and foremost, constructing proofs of correctness demands a clean specification for the problem domain. And of course, a bug in the specification can lead to a bug in the code. So one challenge is finding specifications for systems that can be verified in their own right. Another issue is the cost of developing and maintaining proofs. Originally, we coded proofs by hand. Since then, we have shifted towards a semi-automated style that makes liberal use of custom tactics [11]. The latter approach not only cuts the size of the proofs, but makes them far more robust to changes in the program or specification.

Finally, the programming language embedded in Coq is a relatively high-level, ML like language. For many applications, it is ideal, but for many systems programming tasks (e.g., hypervisors or device drivers), it is too high-level. Thus, we still lack a good low-level programming environment which can effectively replace C.

References

1. PLT Scheme, `http://www.plt-scheme.org/`
2. Leavens, G.T., et al.: Preliminary design of JML: a behavioral interface specification language for Java. SIGSOFT Softw. Eng. Notes 31(3), 1–38 (2006)
3. The Coq Proof Assistant, `http://coq.inria.fr/`
4. Leroy, X.: Formal verification of a realistic compiler. Comm. of the ACM. 52(7), 107–115 (2009)
5. ACL2, `http://userweb.cs.utexas.edu/~moore/acl2/acl2-doc.html`
6. Agda, `http://www.cs.chalmers.se/~catarina/agda/`
7. Epigram, `http://www.e-pig.org/`
8. Isabelle, `http://www.cl.cam.ac.uk/research/hvg/Isabelle/`
9. Nanevski, A., et al.: Polymorphism and separation in Hoare Type Theory. In: 11th ACM Intl. Conf. on Functional Prog, pp. 62–73. ACM Press, New York (2006)
10. Malecha, G., et al.: Towards a verified relational database management system. In: 37th ACM Symp. on Principles of Prog, pp. 237–248. ACM Press, New York (2010)
11. Chlipala, A., et al.: Effective interactive proofs for higher-order imperative programs. In: 14th ACM Intl. Conf. on Functional Prog, pp. 79–90. ACM Press, New York (2009)

Cryptography for Network Security: Failures, Successes and Challenges

Bart Preneel

Katholieke Universiteit Leuven and IBBT
Dept. Electrical Engineering-ESAT/COSIC,
Kasteelpark Arenberg 10 Bus 2446, B-3001 Leuven, Belgium
bart.preneel@esat.kuleuven.be

Abstract. This article discusses the state of the art of cryptographic algorithms as deployed for securing computing networks. While it has been argued that the design of efficient cryptographic algorithms is the "easy" part of securing a large scale network, it seems that very often security problems are identified in algorithms and their implementations.

Keywords: cryptographic algorithms, network security, block ciphers, stream ciphers, MAC algorithms, hash functions.

1 Introduction

The first boom in cryptography can be attributed to the introduction of wireless data communications at the beginning of the 20th century [28]: it is clear that wireless communications are as easy to read for an adversary as for the legitimate receiver. There is also the mistaken perception that intercepting wired communications is really difficult; while the introduction of optical communications has raised the threshold, a well motivated opponent can also bypass this hurdle. From the 1960s, dedicated or switched wired networks were introduced for computer networks. Only military, governmental and financial communications were encrypted; until the early 1990s this encryption was mostly implemented in expensive hardware at the data link layer. The development of the world wide web resulted in broad use of cryptography for e-commerce and business applications. The underlying enabling technologies are inexpensive fast software cryptography and open security protocols such as TLS (SSL), SSH and IPsec as introduced in the second half of the 1990s. In spite of this development, only a small fraction of the Internet traffic is encrypted. Most of this encryption is situated at the network or transport layer; the communication is protected end-to-end (e.g., from the browser in the client to the web server), from gateway to gateway (for a VPN based on IPsec using tunnel mode) or from client to gateway (e.g., a VPN for remote access to company networks). In the last decade we have witnessed an explosion of wireless data networks, including Wireless LANs (WLAN, IEEE 802.11), Personal Area Networks (PANs such as Bluetooth or IEEE 802.15, Zigbee or IEEE 802.15.4, and Ultrawideband or IEEE 802.15.4a)

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 36–54, 2010.

and Wireless Metropolitan Area Networks (WiMAX or IEEE 802.16). All these technologies have been introduced with cryptographic security at the link layer; the early solutions are typically not very robust. In addition mobile data communication is growing on the evolving GSM mobile phones using technologies such as GPRS and EDGE as on the third generation mobiles phones such as 3GSM.

End to end protection of voice communication is a relatively recent phenomenon. The main reason has been technological limitations, but there is also a significant legal barrier, since governments want to maintain the capability to perform wiretaps for law enforcement and national security purposes. Analog voice scramblers do not offer a very high security level. The US delegation in the 1945 Yalta conference brought along very voluminous devices for digital voice encryption; apparently they were never used, a.o. for the poor quality. Efficient digital coding of voice for mass market products arrived in the 1980s: secure digital phones (e.g. the STUs) became available, but outside the government and military environment they were never successful. However, today Voice over IP (VoIP) technologies result in widespread end-to-end security based on software encryption. The first analog mobile phones provided no or very weak security, which resulted in serious embarrassment (e.g., the private conversations of Prince Charles being exposed or the eavesdropping of the Soviet mobile communication systems by the US). The European GSM system designed in the late 1980s provided already much better security, even if many flaws remain; these flaws did not stop the system: in 2010 there are more than 4 billion GSM and WCDMA-HSPA subscribers. The GSM security flaws have been resolved in the 3GSM system, but even there no end-to-end protection is provided. The current generation of smart phones users can clearly run software (such as Skype) with this capability.

This short article tends to briefly describe the situation in terms of cryptographic algorithms used in communication networks. In Sect. 2 we present an update on hash functions, stream ciphers, block ciphers and their modes. Section 3 focuses on public key algorithms and Sect. 4 presents the conclusions.

2 Symmetric Primitives

In this section, we discuss the following symmetric primitives: block ciphers, stream ciphers, MAC algorithms, hash functions and modes for authenticated (or unforgeable) encryption.

2.1 Block Ciphers

Block ciphers are a flexible building block for many cryptographic applications. This includes the original goal of encryption (in CBC, CFB, OFB or CTR mode), but they can also be used to construct MAC algorithms (cf. Sect. 2.3), hash functions (cf. Sect. 2.4), pseudo-random functions and one-way functions.

The DES algorithm was published by the US government in the 1970s; it is a block cipher with a 64-bit block length and a 56-bit key. In spite of initial controversy around its design, the deciding factors in the success of the DES algorithm were the standardization by the US government and the generous licensing conditions. However, in the 1990s it became obvious that the 56-bit key size was no longer adequate.[1] The financial world started moving towards two key triple-DES in the late 1990s; this move was completed around 2006, a few years later than planned. In 2004 NIST (National Institute of Standards and Technology, US) announced that DES was no longer adequate and published a triple-DES specification [72]; two-key triple-DES is approved until 2009, while three-key triple-DES is deemed to be adequate until 2030. The modes for triple-DES have been defined in ANSI X9.52 [2]. The main reason for the limited lifetime of the two-key triple-DES variant is the attack by Wiener and van Oorschot [95] that requires 2^{80} time when 2^{40} known plaintexts are available; this is not a concern for the financial sector, as keys are typically changed frequently and messages are very short. On the other hand, three-key triple-DES is very vulnerable to a related-key attack [58]; in this attack an opponent obtains the encryption of a plaintext P under a key K and a key $K \oplus \Delta$ for a constant Δ. In most contexts such an attack is not feasible, but an exception is applications that use control vectors [68].

In 1997, NIST started an open competition to find a replacement for the DES. The AES algorithm has a block of length of 128 bits, and should support keys lengths of 128, 192 and 256 bits. In October 2000 NIST selected the Rijndael algorithm (designed by the Belgian cryptographers Vincent Rijmen and Joan Daemen) as the AES algorithm [24,39]. In 2003, the US government announced that it would also allow the use of AES for secret data, and even for top secret data; the latter applications require key lengths of 192 or 256 bits. AES is a rather elegant and mathematical design, that among the five finalists offered the best combination of security, performance, efficiency, implementability and flexibility. AES allows for compact implementations on 8-bit smart cards (36 bytes of RAM), but also highly efficient implementations on 32-bit architectures (15 cycles/byte on a Pentium III and 7.6 cycles/byte on a Core 2 [55]). Moreover, hardware implementations of AES offer good trade-offs between size and speed. AES has been taken up quickly by many standards and implementations; in May 2010 more than 1300 AES implementations have been validated by the US government.

So far, AES has resisted all shortcut attacks, including algebraic attacks. In 2009, it was demonstrated by Biryukov and Khovratovich [11] that AES-192 and AES-256 are vulnerable to related-key attacks: the attack on AES-256 requires 4 related keys and 2^{119} encryptions, which is much less than 2^{256}. These attacks indicate that they key schedule of AES should have been stronger; on the other hand, they clearly do not form a practical threat and one can easily defend against them by not allowing any key manipulations or by hashing a key before

[1] A US \$ 1 million machine today would recover a DES key in a few seconds – the same design would have taken 3 hours in 1993 [103].

use. It is also worth to point out that it is not possible to design a cipher that is secure against *any* related key attack.

In 2010 Dunkelman *et al.* [31] have published a related key attack on the 64-bit block cipher KASUMI (that is standardized for GSM under the name A5/3 and that is also used for encryption in 3GPP); the attack requires 4 related keys, 2^{26} plaintexts, 2^{30} bytes of memory and time 2^{32}; while these complexities are rather low, the attack cannot be applied to KASUMI as deployed in current mobile networks.

The most powerful attacks against AES and other block ciphers have not been pure mathematical attacks, but timing attacks based on cache effects – this kind of attack applies in principle to any cryptographic algorithm implementation that uses tables (see e.g. [9,76,94]). This attack is one of the reasons why Intel has decided to add dedicated AES instructions to its processors from 2010 onwards [44]; these instructions also boost the performance of AES to about 0.75 cycles/byte (in decryption mode). Note that the fast implementation of AES of Kašper and Schwabe [55] is bitsliced and hence not vulnerable to cache-based attacks.

2.2 Stream Ciphers

Because of their low implementation cost, additive stream ciphers have been the work horse of symmetric cryptography until the 1980s. They take as input a short secret key and a public initialization value IV and stretch this to a long string that can be simply added to the plaintext to yield the ciphertext. This implies that the encryption transformation is very simple but depends on the location in the plaintext. Hardware oriented stream ciphers typically operate on short data units (bits or bytes) and have a small footprint. The initialization value IV serves for resynchronization purposes. Both the IV and the internal memory need to be sufficiently large to resist time-memory-data tradeoffs (see for example [46,62]).

From the 1960s to the late 1980s, most stream ciphers were based on Linear Feedback Shift Registers (LFSRs) that are optimal for hardware implementations (see for example Rueppel [87] and Menezes *et al.* [71]). However, it has become clear that most LFSR-based stream ciphers are much less secure than expected; powerful new attacks include fast correlation attacks [70] and algebraic attacks [23]. Notable cryptanalytic successes are the attack by Barkan and Biham [3] on A5/1 (the stream cipher used in GSM) and the attack by Lu *et al.* [65] on E0 (the stream cipher used in Bluetooth). Both attacks are realistic attacks on widely used algorithms.

RC4 has been designed in 1987 by Rivest for efficient software encryption on 8-bit machines. RC4 was a trade secret, but leaked out in 1994; it is currently still implemented in browsers (SSL/TLS protocol). While several statistical weaknesses have been identified in RC4 [40,77], the algorithm seems to resist key recovery attacks.

In the last decade, fast stream ciphers have been proposed that are oriented towards 32-bit and 64-bit processors. Two stream ciphers that have been included into the ISO standard are MUGI [100] and SNOW [33]; a strengthened variant of SNOW has been selected as backup algorithm for 3GSM. Between 2004 and 2008 the EU Network of Excellence ECRYPT [32] has organized an open competition eSTREAM with as goal to identify promising stream ciphers that are either very fast in software (128-bit key and 64 or 128-bit IV) or that offer a low footprint in hardware (80-bit key and 32 or 64-bit IV). During the four years of the competition, dozens of stream ciphers have been broken. The competition has resulted in a portfolio with four software-oriented ciphers with a performance of 3-10 cycles/byte (HC-128, Rabbit, Salsa20/12 and Sosemanuk); three hardware-oriented ciphers are recommended (Grain, Mickeyv2, and Trivium). An important conclusion from the eSTREAM project is that for very low footprint implementations, 64-bit block ciphers are more efficient; however, if one desires a very high performance implementation with a low hardware cost, the hardware-oriented stream ciphers offer an improvement with a factor of two to four over block ciphers. More details on the eSTREAM competition can be found in [84].

2.3 Message Authentication Codes (MACs)

Message Authentication Codes are used to authenticate messages between parties that share a secret key. MACs are widely use in networks, because they are more efficient in terms of performance and memory than digital signature schemes. The most widely used constructions are derived from block ciphers or hash functions.

The most popular MAC algorithm for financial transactions is still CBC-MAC. Initially, variants based on DES were used; these have been migrated to triple-DES variants. AES is gradually replacing DES for this application (cf. Sect. 2.1).

The CBC-MAC construction based on an n-bit block cipher can be described as follows. First the input string is padded to a multiple of the block length, and the resulting string is divided into t n-bit blocks x_1 through x_t.

$$c_1 := E_k(x_0) \tag{1}$$

$$c_i := E_k(x_i \oplus c_{i-1}), \quad 1 < i \leq t \ . \tag{2}$$

Here $\oplus$ denotes the bitwise exclusive-or operation. Note that – unlike in CBC encryption – no IV value should be used. The recommended variant for use with DES is the ANSI retail MAC [1]: it computes the MAC value with two independent keys k and k': $\text{MAC}_k(x_0 \ldots x_t) = E_k\left(E_{k'}(c_t)\right)$. For AES, EMAC is the preferred construction: $\text{MAC}_k(x_0 \ldots x_t) = E_{k'}(c_t)$. Here k' is a key derived from k. An even simpler scheme is LMAC; it uses the key k' for the last encryption $(i = t)$.

NIST has published yet another variant under the name of CMAC [73] (CMAC was previously called OMAC [53], which is an optimization of XCBC [14]). CMAC modifies the last computation in CBC-MAC by exoring $k2$ or $k3$ to x_t. The key $k2$ is chosen when the last block x_t requires no padding (i.e., it is of length n), while $k3$ is chosen otherwise. The keys $k2$ and $k3$ are computed as

$k2$ = '2' $\cdot E_k(0^n)$ and $k3$ = '4' $\cdot E_k(0^n)$ where 0^n denotes the n-bit all zero string, '2' and '4' are two elements of the finite field F_{2^n}, and "$\cdot$" represents multiplication in the finite field F_{2^n}.

On the Internet, HMAC is by far the most popular construction [5]; in the light of the attacks on MD4 and MD5 (cf. Sect. 2.4), the HMAC security analysis has been refined by Bellare [4]. The state of the art in cryptanalysis is that HMAC-MD4 has been broken by Leurent *et al.* [41]; their attack requires 2^{88} chosen texts and 2^{95} computations. Some doubts have been cast on HMAC-MD5 [21,59]; the best known attack on HMAC-MD5 is a related key attack that requires 2^{51} chosen plaintexts and 2^{100} time (see also [99]). For the time the security margin offered by HMAC-SHA-1 is acceptable.

In the past five years there has been a growing interest in unconditionally secure MAC algorithms. They were introduced as authentication codes by Simmons [92] and more practical constructions were known as universal hash functions (following Carter and Wegman [101]). If they are combined with a block cipher (such as AES) or a pseudo-random function (such as HMAC), the unconditional security is lost, but they result in MAC algorithms that are very efficient and elegant. UMAC [13] is about 10 times faster than CBC-MAC based on AES or HMAC-SHA-1, but it offers a limited key agility and has a rather large Random Access Memory (RAM) requirement; moreover, Handschuh and Preneel have demonstrated [45] that for a large class of MAC algorithms based on universal hash functions (including UMAC) a few forgeries lead to efficient key recovery. Bernstein's Poly1305-AES [9] is one of the constructions based on polynomial universal hashing. It is only three times faster than AES, but it has a better key agility than UMAC and requires less RAM; it seems also less vulnerable to key recovery attacks.

2.4 Hash Functions

Cryptographic hash functions are a widely deployed primitive for message authentication. They compress strings of arbitrary lengths to strings of fixed lengths (typically between 128 and 256 bits). Cryptographic hash functions need to satisfy the following three security properties [71,79]:

- preimage resistance: it should be hard to find a preimage for a given hash result;
- 2nd preimage resistance: it should be hard to find a 2nd preimage for a given input;
- collision resistance: it should be hard to find two different inputs with the same hash result.

For an ideal hash function with an n-bit result, finding a (2nd) preimage requires approximately 2^n hash function evaluations. On the other hand, finding a collision requires only $2^{n/2}$ hash function evaluations (as a consequence of the birthday paradox). Collision resistance implies 2nd preimage resistance, but the formal relation between these definitions is more complex and subtle than one would expect (see Rogaway and Shrimpton [86]). In practice on requires also other properties such as indifferentiability from a random oracle [22], and pseudo-randomness (this assumes that a secret key is part of the input).

The main application of hash function is digital signature schemes, in which one signs the hash value of a message rather than the message itself. Digital signatures are used in some key establishment protocols to bind a protocol message to an entity. Hash functions can also be used to construct MAC algorithms; the most popular construction of this type is HMAC (cf. Sect. 2.3). HMAC constructions are also used for deriving symmetric keys in protocols such as Diffie-Hellman. In practice HMAC is used with hash functions such as MD5, SHA-1 and RIPEMD-160. In the SSL/TLS protocol, a hash function is used at the end of the handshake protocol (in which the cipher suites are negotiated) to confirm the integrity (TLS version 1.0/1.1 uses the concatenation of MD5 and SHA-1, while in TLS version 1.2 a single hash function is used).

In the last decade, a number of structural weaknesses have been identified in hash functions; these weaknesses are related to the way cryptographic hash functions are constructed from smaller building blocks. Most constructions use a simple iteration, and are therefore called iterated hash functions. The most remarkable attack is a result by Joux [54] who shows that if finding a collision for an iterated hash function takes time T (for an ideally secure hash function $T = 2^{n/2}$), one can find 2^s strings hashing to a single value in time $s \cdot T$. As an example, finding a billion messages that all hash to the same result requires only thirty times the effort to find a single collision. This result has the surprising corollary that the concatenation of two iterated hash functions ($g(x) = h_1(x) \| h_2(x)$) is only as strong as the strongest of the two hash functions (even if both are independent). If h_i is a hash function with an n_i-bit result ($i = 1, 2$ and w.l.o.g. $n_1 \geq n_2$), finding a collision for g requires time at most $n_1 \cdot 2^{n_2/2} + 2^{n_1/2} \ll 2^{(n_1+n_2)/2}$ and finding a preimage or 2nd preimage for g requires time at most $n_1 \cdot 2^{n_2/2} + 2^{n_1} + 2^{n_2} \ll 2^{n_1+n_2}$. If either of the functions is weak, the attacks may work better. This attack is particularly relevant since weaknesses have been discovered in several widely used hash functions (cf. supra) and the concatenation construction has been proposed as a robust solution (e.g. in SSL/TLS). It seems that once the collision resistance of our current iterated hash functions breaks down, the other security properties are also undermined.

Until recently, the most widely used hash functions were MD5 and SHA-1. MD5 is a 128-bit hash function designed by Rivest in 1991 [82]; it is a strengthened version of MD4. MD5 was one of the first cryptographic algorithms that was designed to be fast on 32-bit processors in software. Early cryptanalytic results by den Boer and Bosselaers [26] and Dobbertin [30] indicated that finding collisions for MD5 would require less than 2^{64} operations; in spite of the fact that cryptographers advised against using MD5, the algorithm has been widely deployed. The first collisions for MD5 were announced in 2004 by Wang *et al.* [98], who were able to push the limits on differential attacks by introducing some innovative cryptanalytic techniques; their attack required time 2^{39}, which corresponds to a few hours on a PC. Since then the attack has been further optimized;

the best collision search algorithm known today requires milliseconds [93]. While this represents a major breakthrough, it is important to note that with about US$100 000 of hardware, a brute-force collision search for MD5 (or any 128-bit hash function of comparable cost) should take a few days with the design of van Oorschot and Wiener [96].

In 1995, NIST has published SHA-1 [37]; it is a strengthened version of SHA, which was standardized two years earlier [36] (SHA is now called SHA-0 by some researchers). Both SHA(-0) and SHA-1 have a 160-bit result. While SHA-1 is slower but more secure than MD5, it became very popular for applications that require long term security. In 2005, Wang *et al.* [97] have published a collision search algorithm for SHA-1 that requires only 2^{69} steps, which is 2000 times faster than a brute force collision search. Five years later, several researchers have announced improvements (sometimes even very spectacular ones), but so far none of these attacks has materialized. In 2005 Joux *et al.* [54] found collisions for SHA(-0) with complexity 2^{51}. Today the best collision attack for SHA-0 by Manuel and Peyrin [67] takes only 2^{33} steps. The implications of the attack on SHA(-0) are limited, since this algorithm is not deployed.

The collision attacks on MD4 and MD5 are quite unusual in the sense that they are extremely efficient. However, so far their practical implications have been limited, as very few applications use digital signatures and very few applications require collision resistance. In December 2008, Sotirov et al. [93] created a rogue CA certificate using MD5, which allows them to impersonate any website on the Internet. This attack required cryptanalytic improvements beyond simple collision search. Only after this attack, several Certification Authorities decided to remove MD5 from their offerings. While there is substantial progress with preimage attacks on MD4 and MD5, these attacks are far from practical. Leurent [64] has shown that preimages for MD4 can be found in 2^{102} steps, and the preimage attack by Sasaki and Aoki [89] on MD5 has complexity 2^{123}.

RIPEMD-160 [19] could act as a replacement for SHA-1; it seems to resist all cryptanalytic efforts. NIST has also a series of standards that offer longer hash results: SHA-256, SHA-224, SHA-384 and SHA-512 [38], which are known under the common name SHA-2. Cryptanalysis of the SHA-2 family suggests that this second generation functions has a substantial security margin against collision attacks (the results by Indesteege et al. [48] and Sanadhya and Sarkar [88] can only break 24 out of 64 steps of SHA-256). A third alternative is Whirlpool, a design by Rijmen and Barreto [51] based on the design principles of AES. For the most recent status of attacks on Whirlpool, see [61]. All these hash functions have been standardized by ISO in IS 10118-3 [51], together with SHA-1.

NIST is currently running an open competition for a new hash function standard that will be called SHA-3. Sixty-four submissions have been received, 14 of which are currently being evaluated in the second round. It is expected that NIST will announce the winner by mid 2012. For more details on the SHA-3 competition and on the state of hash functions, see [79].

2.5 Authenticated or Unforgeable Encryption

Most applications need a secure channel between sender and receiver; such a channel requires both confidentiality and data authentication. In the 1980s and 1990s, separate primitives were introduced for each of these properties. However, it is not so hard to show that confidentiality protection without data authentication can lead to serious problems; in particular, such a scheme is vulnerable to a chosen ciphertext attack in which the opponent uses decryption queries to learn information on the plaintext. Practical chosen ciphertext attacks have been demonstrated by several authors; we just mention the attack by Canvel *et al.* on SSL/TLS [20] and the attack by Degabriele and Paterson on IPsec [25].

The first approach to achieve both properties was to introduce redundancy to the plaintext before encryption in order to achieve both goals, but this is clearly not adequate. A first formalization of unforgeable encryption was published by Katz and Yung [56]. Bellare and Namprempre [6] showed that if the MAC algorithm satisfies a strong security requirement (namely strong unforgeability), the best generic solution is to apply a MAC algorithm to the ciphertext (the so-called Encrypt-then-MAC model), which is the option chosen by IPsec. Other alternatives (MAC-then-Encrypt of SSL/TLS and Encrypt and MAC of SSH) can also be shown to be secure, but they require a specific rather than a generic analysis (e.g., taking into account the specific encryption mode).

The above schemes require both an encryption algorithm and a MAC algorithm. Jutla showed that it was possible to achieve both properties at a much lower cost; for this purpose he introduced in 2000 two modes, the IACBC (Integrity-Aware Cipher Block Chaining) and IAPM (Integrity-Aware Parallelizable Mode). Gligor and Donescu proposed the XCBC and XECB schemes in [43]. Rogaway *et al.* [85] introduced an optimized version of IAPM called the OCB mode (Offset CodeBook). These schemes require an overhead of less than 10% over CBC encryption and offer some attractive features; for example, some of them are fully parallellizeable. An important non-technical disadvantage is that all these schemes are encumbered by patents, which has been a barrier to their adoption.

As a consequence of this patent issue, several alternative schemes have been introduced that are slower than these schemes, but that are free. NIST and ISO have standardized a combination of the counter mode with a polynomial based authentication (the Galois Counter Mode or GCM [69,75]) and with CBC-MAC (the Counter with CBC-MAC mode [102,74]). For a more detailed overview of authenticated encryption schemes, see the overview article by Black [12] and the ECRYPT II report [32].

3 Public Key Algorithms

In network security, public key algorithms are only used for the establishment of session keys and for the mutual authentication of the parties. The main reason is that public key operations are two or three orders of magnitude slower than symmetric key primitives. Moreover, the block lengths and overhead are substantially larger. Public key algorithms need to be integrated into a protocol such as the Station-to-Station protocol [29]; more elaborate variants of this protocol have been standardized for SSL/TLS (RFC 5246) and for IPsec (IKEv2 in RFC 4306). The details of these protocols fall outside the scope of this article.

3.1 RSA

RSA, invented by Rivest, Shamir and Adleman in 1978 [83] is by far the most widely used public key algorithm (the RSA patent has expired in 2000). The RSA encryption operation is written as $C = P^e \bmod N$ and the decryption is computed as $P = C^d \bmod N$. Here the encryption and decryption exponent are related by $e \cdot d = 1 \bmod \mathrm{lcm}(p-1, q-1)$, with $N = p \cdot q$. The security of RSA is based on the fact that it is relatively easy to find two large prime numbers p and q, but no efficient methods are known to factor their product N. Note that the security of RSA is based on the fact that extracting random eth roots mod$N$ is hard. This problem could be easier than factoring N (it cannot be harder): surprisingly, whether or not it is easier is still an open problem.

The best known algorithm to factor an RSA modulus N is the General Number Field Sieve (GNFS). Lenstra and Verheul have related the complexity of GNFS to breaking symmetric keys and computing discrete logarithms in [63] (see also the ECRYPT II report on this topic [32]). The current factoring record (achieved in January 2010) is 768 bits [60]. The recommended minimum size for an RSA modulus today is 1024 bits; factoring such a modulus requires approximately 2^{72} steps. Shamir and Tromer [90] proposed in 2003 a hardware design that would need an R&D effort of US$20 M. The hardware cost to factor a 512-bit modulus in ten minutes would be US$ 10 000; a 768-bit modulus could be factored with a similar budget in 95 days; factoring a 1024-bit modulus in 1 year would require a hardware investment of US$ 10 M. Note that these cost estimates do not include the linear algebra step. These estimates show that for long-term security (10-15 years), an RSA modulus of 2048 bits or more is recommended.

Textbook RSA has other weaknesses (see [18] for details). For example, RSA for small arguments is not secure: -1, 0 and 1 are always fixed points and if $P^e < N$ extracting a modular eth root simplifies to extracting a natural eth root, which is an easy problem. In addition, RSA is multiplicative, which means that the product mod$N$ of two ciphertexts will decrypt to the product of the corresponding plaintexts.

The standard PKCS#1v1.5 specifies a padding method for encryption and signing with the RSA algorithm. For encryption, the format consists of the following sequence: a byte equal to 00, a byte equal to 02, at least 8 non-zero padding bytes, a byte 00, and the plaintext. Note that the RSA assumption states that extracting *random* modular eth roots is hard, which means that one should map the plaintext space in a uniform way to the interval $[0, n[$; it is clear that PKCS#1v1.5 is quite far from this goal. This has been exploited by Bleichenbacher [15] to recover the plaintext corresponding to a selected ciphertext using a chosen ciphertext attack (in which encryptions of different but related ciphertexts are obtained); more specifically, Bleichenbacher's attack only needs to know whether the plaintext is of the right format (it is based on the error messages). In 1993, Bellare and Rogaway published the OAEP (Optimal Asymmetric Encryption) transform, together with a security proof [7]. This proof essentially states that if someone can decrypt a challenge ciphertext without knowing the secret key, he can extract random modular eth roots. The proof is in the random oracle model, which means that the hash functions used in the OAEP construction are assumed to be perfectly random. However, seven years later Shoup pointed out that the proof was wrong [91]; the error has been corrected by Fujisaki *et al.* in [42], but the resulting reduction is not very meaningful, that is, the coupling between the two problems is not very tight in this new proof. Moreover, Manger showed that a careful implementation is necessary, since otherwise a chosen ciphertext attack based on error messages may still apply [66]. Currently the cryptographic community believes that the best way of using RSA is the RSA-KEM mode [80]: this is a so-called *hybrid* mode in which RSA is only used to transfer a session key, while the plaintext is encrypted using a symmetric algorithm with this key.

For RSA PKCS#1v1.5 signatures, no practical attack is known, even if this padding format is again very far from random. The RSA signing operation is applied to the following sequence: a byte equal to `00`, a byte equal to `01`, a series of bytes equal to `FF`, a byte `00`, and the hash value (with some ASN.1 prepended). At the rump session of Crypto 2006, Bleichenbacher showed that many implementations of RSA signature verifications stop at the end of the hash value. This opens the possibility to append a large random string S (and shorten the series of `FF` bytes accordingly). It is very easy to choose S such that the complete string is a perfect cube, and extracting cube roots over the integers is easy. This means that one can forge any signature for $e = 3$ without knowing the private key; even better, this forged signature works for any modulus N that is large enough. A variant of the attack is based on the fact that some verification software ignores the content of the ASN.1 string. These attacks can be precluded by implementing a correct verification, which consists of checking that the hash value is right aligned or alternatively by re-generating the whole block as the signer does and checking that it is correct. The problem is however that as a signer may not be able to influence the verification software, hence it is better to increase the verification exponent to $2^{16} + 1$. Implementations that were reported to be vulnerable to this problem include OpenSSL, Mozilla NSS, and GnuTLS. A better solution is to use RSA-PSS [8], which has been included together with OAEP in PKCS#1 v2.1. Even if the scheme dates back to 1996 and the standard to 2002, so far implementors seem to be reluctant to upgrade to the more robust algorithms.

For performance reasons, the RSA private key operations (decryption and signing) are often executed using the Chinese remainder theorem. This means that they are computed $\bmod p$ and $\bmod q$ and that both results are combined to recover the result $\bmod N$. One of the most important vulnerabilities of RSA in practice is the observation by Boneh *et al.* [17]: if a transient fault is introduced in the calculation mod p or mod p (but not both), one can recover p and q. Making an implementation robust against these powerful fault attacks is non-trivial.

An important lesson that can be drawn from this is that it is surprisingly difficult to use RSA correctly: it has taken the cryptographic community more than 20 years to learn how to do this. The most efficient solutions still rely on the random oracle model, and it is an important problem how one can use RSA efficiently without this assumption.

3.2 Elliptic Curve Cryptography (ECC)

Elliptic curve cryptography (ECC) is a public-key primitive that is increasingly important as alternative to RSA. The standards (e.g., [52,47]) support both elliptic curves over F_p with p prime and F_{2^m} with m prime. The first curves can take advantage from an arithmetic coprocessor for RSA if available, while the latter allow for very compact hardware implementations.

An important advantage of elliptic curves are the shorter key lengths. Based on the best known algorithms today, one can estimate that 160-bit elliptic curves correspond to 1248-bit RSA, and 224-bit elliptic curves correspond to 2432-bit RSA (see the ECRYPT II report [32]). For these bit-lengths, signing is about five (resp. 20) times faster with elliptic curves, but verifying a signature is seven (resp. five) times faster with RSA. Moreover, very compact hardware implementations of ECC have been developed.

ECC was proposed in 1985; for the first 15 years the market was reluctant to adopt this new and more complex primitive. However, in the past five years ECC has been selected by the governments of Austria, Germany, Switzerland and the USA and are gaining more widespread acceptance. The main attraction lies clearly in the shorter key lengths; this advantage over RSA will grow larger over time.

4 Conclusions

During the past decade, the AES has become the de facto standard for encrypting network data. HMAC-MD5 and HMAC-SHA-1 are the most common algorithms used for message authentication. We see a gradual evolution towards using mechanisms for authenticated or unforgeable encryption, which combine encryption and data authentication in one operation. Those modes require a redesign of the protocol. In this context, HMAC is increasingly replaced by CBC-MAC based on AES or a polynomial hash function; the latter is substantially faster but perhaps a bit less robust. Wireless networks still use older block ciphers or stream ciphers; 3G networks offer data authentication based on MAC algorithms.

For public key algorithms the evolution has been much slower. RSA and Diffie-Hellman based protocols over F_p are getting more and more competition from ECC, in particular for low footprint or low power environments. The relatively smaller keys for ECC is a key factor in this development.

Side channel attacks have become an important area of research: they currently strongly influence hardware and software implementations, but at the cost of a decreased performance. One can expect that in the future some algorithms will be re-designed from scratch so that implementing these algorithms in a secure way is easier.

In addition to new attacks, new security proofs and models have been developed, that increase our understanding in areas such as modes for confidentiality and authenticated encryption and padding methods for RSA and ECC.

In both cases (new attacks and new models and designs), there is a need for efficient and secure procedures to upgrade and retire cryptographic algorithms. However, even if we live in a world in which the environment can change in days or months, replacing a cryptographic algorithm still takes many years. System designers need to build systems that are agnostic to the cryptographic algorithm and that allow for fast and secure key length and algorithm upgrades.

Acknowledgements. This work was partially funded by the European Commission through the IST Programme under Contract ICT-2007-216676 ECRYPT II and by the Belgian Government through the IUAP Programme under contract P6/26 BCRYPT.

References

1. ANSI X9.19, Financial Institution Retail Message Authentication, American Bankers Association (August 13, 1986)
2. ANSI X9.52, Triple Data Encryption Algorithm Modes of Operation, American Bankers Association (1998)
3. Barkan, E., Biham, E., Keller, N.: Instant ciphertext-only cryptanalysis of GSM encrypted communication. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 600–616. Springer, Heidelberg (2003)
4. Bellare, M.: New proofs for NMAC and HMAC: Security without collision resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
5. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
6. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings ACM Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)
8. Bellare, M., Rogaway, P.: The exact security of digital signatures – How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
9. Bernstein, D.J.: The Poly1305-AES message-authentication code. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 32–49. Springer, Heidelberg (2005)

10. Bernstein, D.J.: Cache-timing attacks on AES (2005) (preprint), http://cr.yp.to/papers.html#cachetiming
11. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
12. Black, J.: Authenticated encryption. In: van Tilborg, H. (ed.) Encyclopedia of Cryptography and Security, pp. 11–21. Springer, Heidelberg (2005)
13. Black, J., Halevi, S., Krawczyk, H., Krovetz, T., Rogaway, P.: UMAC: Fast and secure message authentication. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 216–233. Springer, Heidelberg (1999)
14. Black, J., Rogaway, P.: CBC-MACs for arbitrary length messages. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 197–215. Springer, Heidelberg (2000)
15. Bleichenbacher, D.: Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998)
16. Bleichenbacher, D.: Forging some RSA signatures with pencil and paper. Presented at the Rump Session of Crypto 2006 (2006)
17. Boneh, D., DeMillo, R., Lipton, R.: On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
18. Boneh, D., Joux, A., Nguyen, P.Q.: Why textbook ElGamal and RSA encryption are insecure. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 30–43. Springer, Heidelberg (2000)
19. Bosselaers, A., Dobbertin, H., Preneel, B.: The RIPEMD-160 cryptographic hash function. Dr. Dobb's Journal 22(1), 24–28 (1997)
20. Canvel, B., Hiltgen, A.P., Vaudenay, S., Vuagnoux, M.: Password interception in a SSL/TLS Channel. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 583–599. Springer, Heidelberg (2003)
21. Contini, S., Lin, Y.L.: Forgery and partial key recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
22. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: how to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)
23. Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)
24. Daemen, J., Rijmen, V.: The Design of Rijndael. In: AES – The Advanced Encryption Standard. Springer, Heidelberg (2001)
25. Degabriele, J.P., Paterson, K.G.: Attacking the IPsec standards in encryption only configurations. In: IEEE Symposium on Security and Privacy, pp. 335–349. IEEE, Los Alamitos (2007)
26. den Boer, B., Bosselaers, A.: Collisions for the compression function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
27. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246 (August 2008)
28. Diffie, W., Landau, S.: Privacy on the Line. The Policy of Wiretapping and Encryption, 2nd edn. MIT Press, Cambridge (2007)
29. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Designs, Codes, and Cryptography 2(2), 107–125 (1992)

30. Dobbertin, H.: The status of MD5 after a recent attack. CryptoBytes 2(2), 1–6 (Summer 1996)
31. Dunkelman, O., Keller, N., Shamir, A.: A practical-time attack on the KASUMI cryptosystem used in GSM and 3G telephony. In: Rabin, T. (ed.) Advances in Cryptology, Proceedings Crypto 2010. LNCS. Springer, Heidelberg (2010) (in print)
32. EU Network of Excellence ECRYPT II, Yearly Report on Algorithms and Keysizes (2009-2010), `http://www.ecrypt.eu.org`
33. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 47–61. Springer, Heidelberg (2003)
34. Electronic Frontier Foundation, Cracking DES, Secrets of Encryption Research, Wiretap Politics & Chip Design. O'Reilly & Associates, Sebastopol (1998), Source code of the implementation described in the book can be downloaded from `https://www.cosic.esat.kuleuven.ac.be/des/`
35. EU Directive 1999/93/EC, Community framework for electronic signatures (December 13, 1999)
36. FIPS 180, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180, NIST, U.S. Dept. of Commerce (May 11, 1993)
37. FIPS 180-1, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-1, NIST, U.S. Dept. of Commerce (April 17, 1995)
38. FIPS 180-2, Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-2, NIST U.S. Dept. of Commerce (August 26, 2002) (Change notice 1 published on December 1, 2003)
39. FIPS 197, Advanced Encryption Standard, Federal Information Processing Standard, NIST, U.S. Dept. of Commerce (November 26, 2001)
40. Fluhrer, S., Mantin, I., Shamir, A.: Weaknesses in the key scheduling algorithm of RC4. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 1–24. Springer, Heidelberg (2001)
41. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
42. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 260–274. Springer, Heidelberg (2001)
43. Gligor, V.D., Donescu, P.: Fast encryption and authentication: XCBC encryption and XECB authentication modes. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 92–108. Springer, Heidelberg (2002)
44. Gueron, S.: Intel's new AES instructions for enhanced performance and security. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 51–66. Springer, Heidelberg (2009)
45. Handschuh, H., Preneel, B.: Key-recovery attacks on universal hash function based MAC algorithms. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 144–161. Springer, Heidelberg (2008)
46. Hong, J., Sarkar, P.: New applications of time memory data tradeoffs. In: Roy, B.K. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 353–372. Springer, Heidelberg (2005)
47. IEEE P1363, Standard Specifications for Public Key Cryptography (2000)
48. Indesteege, S., Mendel, F., Preneel, B., Rechberger, C.: Collisions and other nonrandom properties for step-reduced SHA-256. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 276–293. Springer, Heidelberg (2009)

49. ISO/IEC 7816, Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange (1997)
50. ISO/IEC 9797, Information technology – Security techniques – Message Authentication Codes (MACs), Part 1: Mechanisms using a block cipher, 1999, Part 2: Mechanisms using a hash-function (2002)
51. ISO/IEC 10118, Information technology – Security techniques – Hash-functions, Part 1: General (2000), Part 2: Hash-functions using an n-bit block cipher algorithm (2000), Part 3: Dedicated hash-functions (2004), Part 4: Hash-functions using modular arithmetic (1998)
52. ISO/IEC 14888-3, Information technology – Security techniques – Digital signatures with appendix, Part 3: Certificate-based mechanisms (2006)
53. Iwata, T., Kurosawa, K.: OMAC: One key CBC MAC. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 129–153. Springer, Heidelberg (2003)
54. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
55. Käsper, E., Schwabe, P.: Faster and timing-attack resistant AES-GCM. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 1–17. Springer, Heidelberg (2009)
56. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 284–299. Springer, Heidelberg (2001)
57. Kaufman, C.: Internet Key Exchange (IKEv2) Protocol, RFC 4306 (December 2005)
58. Kelsey, J., Schneier, B., Wagner, D.: Key-schedule cryptoanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 237–251. Springer, Heidelberg (1996)
59. Kim, J., Biryukov, A., Preneel, B., Hong, S.: On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1 security in communication networks. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
60. Kleinjung, T., Aoki, K., Franke, J., Lenstra, A.K., Thomé, E., Bos, J.W., Gaudry, P., Kruppa, A., Montgomery, P.L., Osvik, D.A., te Riele, H., Timofeev, A., Zimmerman, P.: Factorization of a 768-bit RSA modulus, Advances in Cryptology. In: Rabin, T. (ed.) Advances in Cryptology, Proceedings Crypto 2010. LNCS, Springer, Heidelberg (2010)
61. Lamberger, M., Mendel, F., Rechberger, C., Rijmen, V., Schläffer, M.: Rebound distinguishers: results on the full Whirlpool compression function. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 126–143. Springer, Heidelberg (2009)
62. Lano, J.: Cryptanalysis and Design of Synchronous Stream Ciphers, PhD Thesis, COSIC, K.U.Leuven (June 2006)
63. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. J. Cryptology 14(4), 255–293 (2001)
64. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
65. Lu, Y., Meier, W., Vaudenay, S.: The conditional correlation attack: A practical attack on Bluetooth encryption. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 97–117. Springer, Heidelberg (2005)

66. Manger, J.: A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS#1 v2.0. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 230–238. Springer, Heidelberg (2001)
67. Manuel, S., Peyrin, T.: Collisions on SHA-0 in one hour. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 16–35. Springer, Heidelberg (2008)
68. Matyas, S.M.: Key Processing with Control Vectors. J. Cryptology 3(2), 113–136 (1991)
69. McGrew, D., Viega, J.: The security and performance of the Galois/Counter Mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004), Full paper http://eprint.iacr.org/2004/193/
70. Meier, W., Staffelbach, O.: Fast correlation attacks on stream ciphers. J. Cryptology 1(3), 159–176 (1989)
71. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
72. NIST Special Publication 800-67, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher (May 2004)
73. NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication (May 2005)
74. NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality (May 2004)
75. NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC (November 2007)
76. Osvik, D., Shamir, A., Tromer, E.: Cache attacks and countermeasures: The case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006), Extended version at, www.wisdom.weizmann.ac.il/~tromer/papers/cache.pdf
77. Paul, S., Preneel, B.: Analysis of non-fortuitous predictive states of the RC4 key stream Generator. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 30–47. Springer, Heidelberg (2003)
78. Petrank, E., Rackoff, C.: CBC MAC for real-time data sources. J. Cryptology 13(3), 315–338 (2000)
79. Preneel, B.: The first 30 years of cryptographic hash functions and the NIST SHA-3 competition. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 1–14. Springer, Heidelberg (2010)
80. Preneel, B., Biryukov, A., De Cannière, C., Örs, S.B., Oswald, E., Van Rompay, B., Granboulan, L., Dottax, E., Martinet, G., Murphy, S., Dent, A., Shipsey, R., Swart, C., White, J., Dichtl, M., Pyka, S., Schafheutle, M., Serf, P., Biham, E., Barkan, E., Braziler, Y., Dunkelman, O., Furman, V., Kenigsberg, D., Stolin, J., Quisquater, J.-J., Ciet, M., Sica, F., Raddum, H., Knudsen, L., Parker, M.: Final report of NESSIE, New European Schemes for Signatures, Integrity, and Encryption. LNCS. Springer, Heidelberg (in print)
81. Preneel, B., van Oorschot, P.C.: MDx-MAC and building fast MACs from hash functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
82. Rivest, R.L.: The MD5 message-digest algorithm, RFC 1321 (April 1992)
83. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications ACM 21(2), 120–126 (1978)
84. Robshaw, M.J.B., Billet, O. (eds.): New Stream Cipher Designs. LNCS, vol. 4986. Springer, Heidelberg (2008)

85. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A block-cipher mode of operation for efficient authenticated encryption. In: ACM Conference on Computer and Communications Security, pp. 195–205. ACM Press, New York (2001)
86. Rogaway, P., Shrimpton, T.: Cryptographic hash function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy, B.K., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
87. Rueppel, R.A.: Analysis and Design of Stream Ciphers. Springer, Heidelberg (1986)
88. Sanadhya, S.K., Sarkar, P.: New collision attacks against up to 24-step SHA-2. In: Roy Chowdhury, D., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 91–103. Springer, Heidelberg (2008)
89. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2010)
90. Shamir, A., Tromer, E.: Factoring large numbers with the TWIRL device. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 1–26. Springer, Heidelberg (2003)
91. Shoup, V.: OAEP reconsidered. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 239–259. Springer, Heidelberg (2001)
92. Simmons, G.J. (ed.): Contemporary Cryptology: The Science of Information Integrity. IEEE Press, Los Alamitos (1991)
93. Sotirov, A., Stevens, M., Appelbaum, J., Lenstra, A.K., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 55–69. Springer, Heidelberg (2009)
94. Tsunoo, Y., Saito, T., Suzaki, T., Shigeri, M., Miyauchi, H.: Cryptanalysis of DES implemented on computers with cache. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 62–76. Springer, Heidelberg (2003)
95. van Oorschot, P.C., Wiener, M.J.: A known plaintext attack on two-key triple encryption. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 318–325. Springer, Heidelberg (1991)
96. van Oorschot, P.C., Wiener, M.: Parallel collision search with cryptanalytic applications. J. Cryptology 12(1), 1–28 (1999)
97. Wang, X., Lin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
98. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
99. Wang, X., Yu, H., Wang, W., Zhang, H., Zhan, T.: Cryptanalysis on HMAC/NMACMD5 and MD5-MAC. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 121–133. Springer, Heidelberg (2010)
100. Watanabe, D., Furuya, S., Yoshida, H., Takaragi, K., Preneel, B.: A new keystream generator MUGI. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 179–194. Springer, Heidelberg (2002)
101. Wegman, M.N., Carter, J.L.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences 22(3), 265–279 (1981)

102. Whiting, D., Housley, R., Fergnson, N.: Counter with CBC-MAC (CCM), RFC 3610 (September 2003)
103. Wiener, M.J.: Efficient DES key search. Presented at the Rump Session of Crypto 1993 (1993); Stallings, W. (ed.): Reprinted in Practical Cryptography for Data Internetworks, pp. 31–79. IEEE Computer Society, Los Alamitos (1996)
104. Yu, H., Wang, G., Zhang, G., Wang, X.: The Second-Preimage Attack on MD4. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 1–12. Springer, Heidelberg (2005)

Group-Centric Models for Secure and Agile Information Sharing

Ravi Sandhu[1,2], Ram Krishnan[1],
Jianwei Niu[1,2], and William H. Winsborough[1,2]

[1] Institute for Cyber Security
[2] Department of Computer Science
University of Texas at San Antonio
{ravi.sandhu,ram.krishnan}@utsa.edu,
niu@cs.utsa.edu,wwinsborough@acm.org

Abstract. To share information and retain control (share-but-protect) is a classic cyber security problem for which effective solutions continue to be elusive. Where the patterns of sharing are well defined and slow to change it is reasonable to apply the traditional access control models of lattice-based, role-based and attribute-based access control, along with discretionary authorization for further fine-grained control as required. Proprietary and standard rights markup languages have been developed to control what a legitimate recipient can do with the received information including control over its further discretionary dissemination. This dissemination-centric approach offers considerable flexibility in terms of controlling a particular information object with respect to already defined attributes of users, subjects and objects. However, it has many of the same or similar problems that discretionary access control manifests relative to role-based access control. In particular specifying information sharing patterns beyond those supported by currently defined authorization attributes is cumbersome or infeasible. Recently a novel mode of information sharing called group-centric was introduced by these authors. Group-centric secure information sharing (g-SIS) is designed to be agile and accommodate ad hoc patterns of information sharing. In this paper we review g-SIS models, discuss their relationship with traditional access control models and demonstrate their agility relative to these.

Keywords: DAC, Groups, LBAC, MAC, RBAC, Secure Information Sharing.

1 Introduction

The need to *share but protect* is one of the oldest and most challenging problems for trustworthy computing. Saltzer-Schroeder [1] identified the desirability and difficulty of maintaining "some control over the user of the information even after it has been released." The ensuing three and half decades have further compounded the technical difficulties to the point where one may ask if it is even reasonable to seek solutions. The analog hole [2] wherein content is captured at the point it is rendered into human perceptible form and converted back into unprotected digital form highlights the intrinsic limits. At the same time our increasingly information-rich and information-dependent

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 55–69, 2010.

society needs to exploit *secure information sharing* (SIS) to fully benefit from the productivity, social and national security benefits of the ongoing cyber revolution.

SIS presents two major research challenges. The *containment challenge* is to ensure that protected information is accessible on the recipient's computer only as permitted by policy, including inability to make unprotected or less-protected copies. The latter has inherent limits such as the analog hole. Containment requires a trusted computing base on the recipient's machine and a mix of cryptography and access control, with the degree of assurance correlated with tamper-resistance. There is a rich literature on containment including the currently dominant TCG approach [3]. While high assurance is elusive and may remain so, there is consensus that low to medium assurance is within state-of-the-art.

In this paper, we assume that adequate assurance for containment is available commensurate with the application. We focus on the *policy challenge* of specifying, analyzing and enforcing SIS policies assuming adequate containment. A basic premise is that this requires new access control models that can integrate and go beyond earlier ones, have intuitive grounding and rigorous mathematical foundations, are usable by the ordinary citizen and enforceable in distributed systems. The paper will build upon a novel approach called Group-centric Secure Information Sharing (g-SIS) recently introduced by the authors [4,5,6]. Another basic premise is that the policy challenge in specifying and analyzing the intrinsic application policy should be clearly separated from enforcement policy issues that arise due to the realities and practicalities of a distributed system. Following [7,8,9] we call these respectively P-layer (for application policy) and E-layer (for enforcement policy) concerns. These premises are elaborated below.

Although many access control models have been published and analyzed, only three have received meaningful practical traction [10]. Discretionary access control (DAC) [11,12,13] enforces controls on sharing information at the discretion of the "owner" of the information but fails containment completely by allowing unprotected copies to be made. (Originator Control or ORCON [14,15,16,17] attaches policies from the original to the copies to fix this defect, but does not directly address the policy challenge.) Lattice-based access control (LBAC) [11,18,19,20] restricts information to flow in one direction in a lattice of security labels. Copies inherit the least upper bound of labels from the originals and remain contained. Information sharing in LBAC is essentially preordained in that information is either not shared or shared with everyone who has a sufficiently strong clearance. Any deviation from this pattern requires creation of a new label, which is not supported in existing LBAC models and breaks their existing mathematical foundations. Role-based access control (RBAC) [21,22] is designed to facilitate assigning permissions based on job function and such considerations. Although RBAC can be configured to enforce DAC and LBAC [23] it is not designed with information sharing in mind, so it does not directly address the containment or policy challenges. (Attribute-based access control models such as UCON [24] and XACML [25] use general attributes in addition to roles and security labels, but likewise do not directly address containment or policy.) This bears out the premise that new access control models are needed for SIS. At the same time these successful classic models embody intuitions and principles that are likely to be vital to a comprehensive solution.

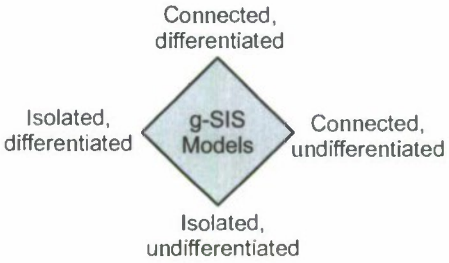

Connected, differentiated
Isolated, differentiated
g-SIS Models
Connected, undifferentiated
Isolated, undifferentiated

Fig. 1. A family of g-SIS models

The premise of sharply separating P- and E-layers builds on the much practised policy/mechanism separation principle first articulated in HYDRA [26]. P-layer specifications express a policy that is ideal in the sense that it ignores issues such as distributed authorization state, network latency, caching, and requirements for off-line use. E-layer specifications define authorization decisions that approximate those given by the ideal policy in a manner that provides the desired application-dependent balance between resource availability and timely propagation of authorization-state changes. They also include additional entities such as trusted authorization/revocation servers which are abstracted out at the P-layer.

This paper primarily focusses on P-layer aspects of g-SIS. In g-SIS, users and information come together in a group to facilitate sharing. Users gain access to group information by virtue of membership. Likewise information is made available to members by adding it to the group. Constituting a group as the unit of SIS provides many of the same benefits of using roles versus individual users for permission distribution. Two useful metaphors for a g-SIS group are a subscription service and a secure meeting room. Subscription disseminates information to subscribers who participate in blogs and forums. A meeting room brings people together to share information available in the room. The times at which users join and leave and at which objects are added and removed affect user authorizations both during and after periods of group membership. For example, in the much studied secure multicast problem [27] new members joining the group cannot access content added prior to joining (backward secrecy) and members leaving the group cannot access new content thereafter (forward secrecy). The requirements of a committee meeting room could allow members access to older information once they join (no backward secrecy). These metaphors further indicate the need for multiple groups. In the simplest case we can have multiple groups that are *isolated* or *independent* in that membership in one group has no impact on what a user can do in another group, whereas with *coupled* or *connected* groups such impact can occur. A theory of g-SIS thus needs to model and enable specification of such temporal and coupling interactions. Looking within a group we can distinguish *undifferentiated* versus *differentiated* groups. In an undifferentiated group user authorizations are undifferentiated once users are admitted into the group. Specifically, authorizations do not depend on attributes other than group membership (and associated temporal relations between users and objects as discussed above earlier). Combining these two characteristics of groups we have four possible cases shown in figure 1 for g-SIS models. In this figure the lowest class (isolated, undifferentiated) is included in all the higher classes; the highest class (connected, differentiated) includes all the others; and the two classes in the middle (isolated, differentiated) and (connected, undifferentiated) are incomparable in this respect.

Our prior work [5] primarily focussed on the isolated group model. In this paper, we outline our vision on building the connected, undifferentiated group model and compare it with classic access control models such as LBAC, Domain and Type Enforcement [28] and RBAC. We show that our proposed connected, undifferentiated group model can express such policies and conveniently handle more dynamic information sharing scenarios. The remainder of this paper is organized as follows. In section 2, we briefly review the isolated group model. In section 3, we discuss candidate intergroup relationships for the connected group model. We also discuss constructions of LBAC [20] and a read-write $RBAC_0$ model [22] and demonstrate the agility of the connected group model in relation to these. We conclude in section 4.

2 Background

Group-Centric models for secure information sharing (g-SIS) have been recently introduced [4,5,6]. In this paper we focus entirely on undifferentiated groups. There are then two classes of g-SIS models: isolated, undifferentiated (g-SIS^i) and connected, undifferentiated (g-SIS^c). For convenience we will henceforth drop explicit mention of undifferentiated and simply call these two classes isolated and connected respectively. In g-SIS^i, groups are isolated in the sense that they do not directly interact with each other. For instance, a user's membership in one group has no implication on her authorizations in other groups. Our prior work [4,5,6] focusses primarily on isolated g-SIS models. In g-SIS^c, groups may be related. For instance, user's membership in one group may be contingent upon her membership in another group or groups could be hierarchical where users in one group may dominate another group. In this section, we briefly review the core aspects of isolated g-SIS models. In the subsequent sections, we discuss candidate relationships in the connected group models.

In g-SIS^i, a group is established, for instance, between two or more organizations for a specific purpose. Users from these organizations may join, leave and possibly re-join the group. Similarly, objects from participating organizations may be added, removed and possibly re-added. Users in the group may read and write such group objects and potentially create new objects in the group. Such new objects typically represent intellectual property created as a result of collaboration between participating organizations. In such scenarios, authorizations in the group may depend upon various aspects such as the time at which a user joined and the time at which the object was added. Specifically, there is a requirement of simultaneous membership of a user and an object in order to be able to read/write the object.

g-SIS^i recognizes a range of group policies. For instance, in some scenarios, users may be authorized to access certain objects even after leaving the group. In another, a joining user may access objects added prior to her join time. Two metaphors highlight such scenarios: secure meeting room and subscription service. For the secure meeting room metaphor, consider a program committee meeting where participants discuss in a room. Suppose, Alice is a member whose paper is currently discussed. Typically, Alice steps out of the room for a brief period. During this period, Alice may retain access to discussions that occurred prior to the time at which she left the room. Further, on re-joining the room at a later period, her access to discussions resumes (except those that occurred during her period of absence). In another scenario (where Alice to had to step out of the room for reasons other than conflict-of-interest), discussions that occurred during Alice's absence may be recorded in a white board and she may access them on re-join.

For the subscription service metaphor, consider a secure multicast network which typically has a notion of backward and forward secrecy. When a node joins the multicast network, it cannot access data distributed on the network prior to join time (backward secrecy). When a node leaves the network, it cannot access data shared between other nodes after leave time (forward secrecy).

In general, there could be numerous variations of such policies in g-SIS^i. A g-SIS^i *specification* characterizes the precise conditions under which a user is authorized to perform a certain action (such read and write) on an object. All g-SIS^i specifications are required to satisfy a set of *core properties*. The core properties specify under what conditions it is appropriate for a specification to hold in the g-SIS^i model. We informally discuss these properties below (see [5] for a formal treatment).

Persistence Properties: This class of properties specifies that authorization may not change unless some authorization changing event occurs. In g-SIS^i, authorization changing events include a user joining and leaving a group and an object being added and removed from a group. Authorization (or Revocation) persistence property states that if a user is authorized (or not authorized) to access an object in a group, she will remain so unless one of the authorization changing event occurs.

Authorization Provenance: This class of properties is concerned about when authorization may begin to hold. As mentioned earlier, in certain scenarios, it is possible that a user may be able to access a group object even after leaving the group. (For instance, after the subscription ends, the user may retain access to articles that she had paid for.) This property states that a user's authorization to access an object may begin to hold for the *first time* only after a simultaneous period of group membership between the user and the object in question. Note that subsequent times at which the same authorization holds have no such requirement. Thus it is possible to construct a valid g-SIS^i specification in which after an initial overlapping period of user and object membership, the user may continue to remain authorized for that object even after leaving the group (or even after the object is removed from the group).

Bounded Authorization: This class of properties is concerned about what authorizations are allowed to hold during the non-membership periods of users/objects. For users, the property states that the set of objects that a user is authorized to access after she leaves the group cannot increase after leave time. (Note that she may lose access to such objects after leave time but she cannot gain access to new objects after leaving the group.) Similarly, for objects, the property states that the set of users authorized to access an object after it is removed from the group cannot increase after remove time.

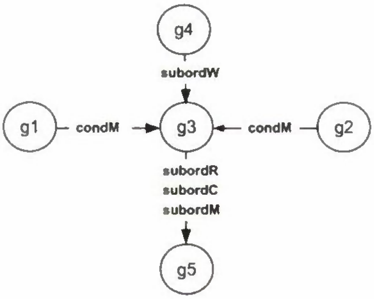


Fig. 2. A snapshot of relationships between various groups

In [5], we characterized a variety of useful authorization semantics for user join and leave operations and object add and remove operations. For instance, a *Strict* join to a group restricts a user's access to objects added to the group after join join time while a *Liberal* join allows access to all objects. We also developed a family of g-SIS^i specifications based on such authorization semantics and showed that they satisfy the core properties. In our follow on work, we have also shown that the core properties are logically consistent and mutually independent. We have further considered additional core properties in light of versioning support for object write. Here each object is composed of a growing set of versions and any specific version may be written to create a new version. Further, the core properties accommodate additional authorization changing operations such as update and object create.

3 Connected Group g-SIS Models

In this section, we introduce a connected g-SIS model (g-SIS^c) where groups are connected by some type of relationship. Before we discuss these relationships, it is important we distinguish the notion of user from that of a subject in access control. Typically, user a representation of a human being in the system (e.g. user id) and subjects represent processes (e.g. a word processing program) that a user may create to carry out various tasks. A user is typically trusted, within limits, in the system while a subject is not. For instance, a subject may be a trojan horse performing some hidden malicious activities such as a word processing program uploading contents to a remote server. Thus a user may create a subject with restricted privileges for containment purposes.

3.1 Inter-group Relationship Semantics in g-SIS^c

We discuss a few candidate inter-group relationships for the g-SIS^c model below:

1. Conditional Membership (condM): A conditional membership relation between two groups specifies that a users membership in one group is contingent upon her membership in another group. We define conditional membership relation to be reflexive. Transitivity and symmetry must be explicitly defined if required. Conditional membership requirements are common in collaboration scenarios. For instance, consider a collaboration group g3 established between two organizations

represented by groups g1 and g2 respeetively (sce figure 2). It is typieal that every user in g3 is requircd to be a member of either g1 or g2. The definitions condM(g3,g1) and condM(g3,g2) ean easily speeify this requirement. Note that conditional membership is a relation defined between groups for users. It does not specify any direet requirement on subjects.

2. Subordination: The subordination relations, in general, characterize the notion of one entity dominating another. In g-SIS^c, we define a number of subordination relations where one group dominates another in different ways. Again, all of these relationships are reflexive by definition. Transitivity and symmetry must be explicitly defined if required.
   - Create Subordination (subordC): A subordC(g3,g5) definition states that users in group g3 may create subjects in group g5.
   - Read Subordination (subordR): A subordR(g3,g5) definition states that subjects in group g3 may read objects in group g5.
   - Write Subordination (subordW): A subordW(g4,g3) definition states that subjeets in group g4 may write to objeets in group g3.
   - Move Subordination (subordM): A subordM(g3,g5) definition states that subjects in group g3 may move to group g5. After moving to g5, the subject no longer resides in g3 which may result in losing access to objects in g3.

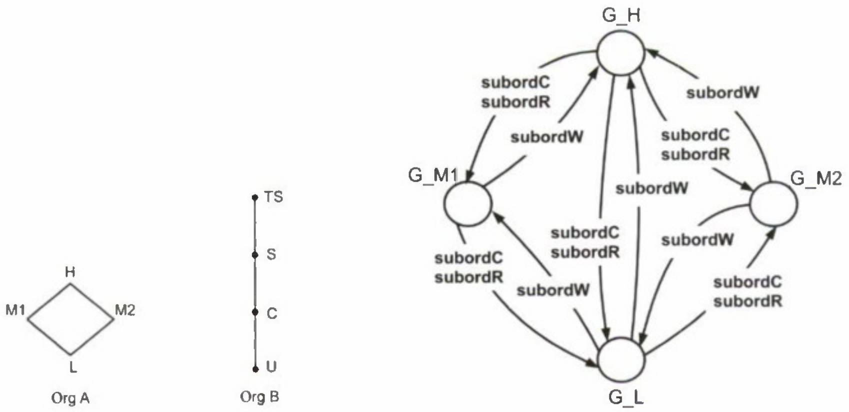
   Evidently, these subordination relations allow users in one group to read and write objects in another related group by means of their subjeets.
3. Mutual Exclusion: Two groups may be specified to be mutual exclusive with respect to membership. That is a user (or an objeet) may not be a member of mutually exclusive groups at the same time. Furthermore, dynamic mutual exclusion can also be specified where a user may be a member of two mutually exclusive groups but cannot ereate subjects in the two groups at the same time.
4. Cardinality: There could be many different types of eardinality eonstraints. For instance, a group could have membership cardinality for users, subjects and objects. Furthermore, a cardinality restriction on the number of relationships that a group may have with other groups could be specified.

Figure 2 shows a snapshot of relationships established between different groups. An important aspect of g-SIS^c is that relationships may change over time as per the varying requirements of the information sharing or collaboration application.

3.2 Configuring LBAC Policies in g-SIS^c

In this section, we diseuss how Lattice-Based Access Control [20] policies such as Bell-LaPadula [18] information flow policies can be easily configured using the relationships defined in g-SIS^c. We also demonstrate the agility of g-SIS^c by shoing how it addresses some of the limitations of LBAC models.

Figure 3(a) shows two sample Bell-LaPadula lattiees for orgs A and B. The org A lattice has four security labels: L, M1, M2 and H. In LBAC, the domination relationship is reflexive, transitive and anti-symmetric. In this lattice, M1 and M2 dominate L and H dominates M1 and M2 (and L by transitivity). M1 and M2 are ineomparable. As per standard terminology, users are assigned one of these four security *clearances* and



(a) A sample lattice for one-directional information flow.

(b) An equivalent g-SIS^c configuration of org A lattice.

Fig. 3. LBAC in g-SIS^c

objects are assigned one of these four *classifications*. Users may then create a subject with a clearance that is dominated by the user's clearance. A subject may read objects whose classifications are dominated by the subject's clearance. A subject may write to objects whose classifications dominate the subject's security clearance.

Figure 3(b) shows an equivalent construction of org A lattice in g-SIS^c. It consists of four groups G_L, G_M1, G_M2 and G_H representing the labels L, M1, M2 and H respectively. Read, write and subject create subordination relationships have been defined according to the specification of the org A lattice in figure 3(a). The subordination relationships are defined in such a manner that a group at the arrow end is subordinate to the group at the tail end. For instance, G_M1 is both create subject and read subordinate to G_H, while G_H is write subordinate to G_M1. Since the relationships are not transitive, we needed to define direct subordination relationships between G_H and G_L as shown in the figure.

Suppose orgs A and B in figure 3(a) need to collaborate on a mission. Specifically, suppose that org B wants to share all its S classified objects (but not its TS and C classified objects) with H cleared users in org A. This is not feasible by simple adjustments to the two lattices in figure 3(a).

Figure 4 shows a construction in g-SIS^c that allows such collaboration scenarios. By assigning a read subordination relation between groups G_H and G_S and groups G_H and G_C respectively, org B is able to allow H cleared org A users to read both S and C classified org B objects. If the subordRrelation is excluded between G_H and G_C, read access can be restricted to S cleared objects.[1] Note that other types of subordination relationships may be specified between org A groups and org B groups to realize

[1] It is true that information may flow from G_C to G_S and thus restricting org A users' access only to G_S may not be completely feasible. Nevertheless, only information that is explicitly copied from G_C to G_S by a subject is available to G_H users. G_H users do not have direct access to G_C objects.

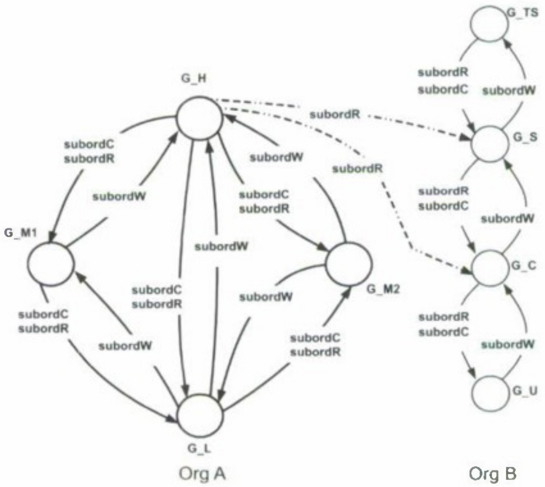


Fig. 4. Agile collaboration enabled by g-SIS^c

other interesting policies. For instance, G_L users may be allowed to write to G_TS objects by defining subordW(G_L, G_TS). These relationships are temporary and may be terminated or modified as collaboration evolves.

Now consider another collaboration scenario illustrated in figure 5. Suppose org A and org B need to collaborate on a mission. They establish groups G1 and G2. TS users/objects from org A and H users/objects from org B may join/be added to G1 (similarly for G2). Conditional membership relations between groups TS and G1 and groups H and G1 are respectively defined. This ensures that if a user leaves the source organization, her membership in G1/G2 is automatically terminated. New information may be created in G1 and G2 as a result of collaboration whieh may be exported to groups E1 and E2 respectively. The export operation may be performed only by special subjects that have administrative rights in the system. By defining a subordRrelationship between respective source organization groups and these export groups, we allow periodic updates about the mission to be communicated to users in source organizations.

3.3 Configuring Domain and Type Enforcement in g-SIS^c

Domain and Type Enforcement (DTE) (see [28] for example) assigns a subject to a specific domain and an object to a specific type and enforces information flow by specifying the read and write permissions in the form of a matrix. A elassic example of the application of DTE is to address the problem of trusted pipelines. Suppose org A (figure 3(a)) needs to enforce that information may flow from L to H but only via M1 or M2.[2] This is not possible to achieve in classic LBAC. Due to the transitive nature

[2] For instance, before a subject at some clearance level may write to a print queue, the document needs to be sent to a trusted print queue manager that visibly stamps every page of the document to be printed with the correct label. In this scenario, the subject should not bypass the queue manager and write to the printer directly.

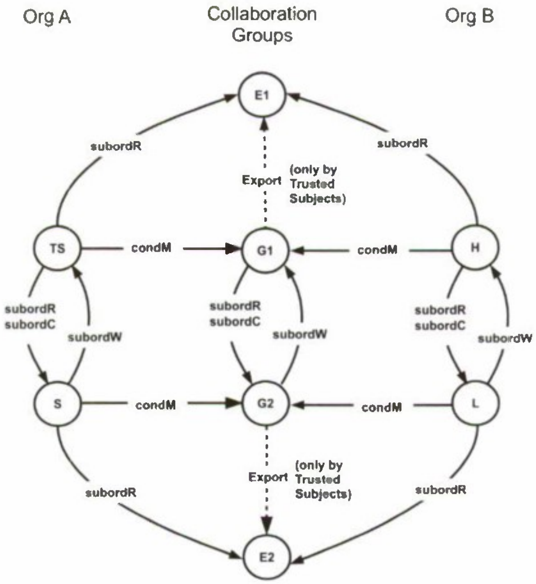


Fig. 5. A collaboration scenario between orgs A and B. The four groups in the middle column (G1, G2, E1 and E2) are established for collaboration between org A (groups in first column) and org B (groups in third column). Groups E1 and E2 are used for exporting new information created as a result of collaboration to G1 and G2 respectively. As indicated, the export operation may be performed only by trusted/administrative subjects.

Objects →

Subjects ↓

Type Domain	H_Ty	M1_Ty	M2_Ty	L_Ty
H_Dom	rw	r	r	r
M1_Dom	w	rw	-	r
M2_Dom	w	-	rw	r
L_Dom	-	w	w	rw

Fig. 6. A DTE matrix to enforce a trusted pipeline from L to H via M1 or M2 for org A lattice in figure 3(a). Note that a subject in L_Dom cannot write directly to objects in H_Ty.

of domination relation, subjects in L may directly write to objects in H (bypassing M1 and M2). In order to achieve this, DTE assigns subjects to domains (instead of security clearances) and objects to types (instead of classifications) and specifies the rights in the form a matrix as shown in figure 6. Note that, as per this matrix, a subject in L_Dom cannot directly write to H_Ty. However, L_Dom subjects may write, for instance, to M1_Ty and M1_Dom subjects may then read that object and write to H_Ty.

Figure 7 shows an equivalent g-SIS^c configuration for the DTE matrix in figure 6. Users join one of the four first level of groups (H_G, M1_G, M2_G and L_G). The second level of groups represent domains for subjects. A user in one of the first level groups

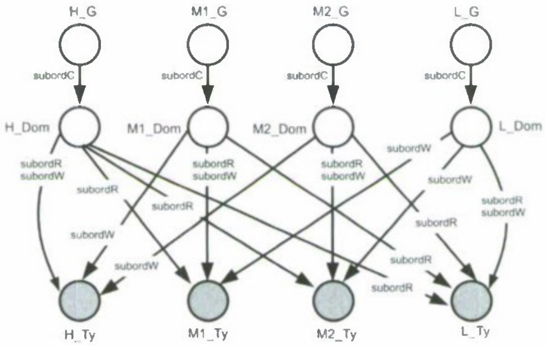


Fig. 7. An equivalent g-SIS^c configuration of the DTE matrix in figure 6. Users join one of the first level of groups (light gray). Users may create subjects in the second level groups representing domains. Objects belong to the third level groups (dark gray) representing types.

may create a subject in the second level domain groups as per the create subject subordination relation (subordC) defined between them. The third level of groups represent the types for objects. Read and write subordination relations are defined between the domain and type groups as per the DTE matrix in figure 6.[3]

3.4 Configuring RBAC Policies in g-SIS^c

In this section, we show the configuration of Role-Based Access Control (RBAC) models [22] in g-SIS^c. In RBAC, a set of roles are created which typically represent job functions of users (employees) in an organization. Each role is assigned with a set of abstract permissions (permission-role assignment) such as credit and debit and users are assigned to specific roles (user-role assignment). Users may activate any combination of roles assigned to them by creating a session. Sessions in RBAC are similar to subjects. The permissions available to a user in a session is the set of all permissions assigned to the set of roles activated in the session by the user. Users may dynamically activate and de-activate specific roles in the session for containment purposes. A family of models have been specified in the well-known RBAC96 [22]. RBAC$_0$ is the basic model described above. RBAC$_1$ supports role hierarchies (where a role inherits the permissions of other roles that it dominates). RBAC$_2$ supports constraints such as separation of duty and role cardinality. RBAC$_3$ supports all the features of RBAC$_0$, RBAC$_1$ and RBAC$_2$ models.

Here we only discuss the basic model, RBAC$_0$. g-SIS^c is a model for information sharing where read and write permissions to objects are of concern. However RBAC supports abstract permissions (to accommodate varied permissions in an organization) and hence it is not feasible to directly configure RBAC policies in g-SIS^c. For the

[3] The figure outlines the approach for the construction but excludes some finer details. For instance, users/objects may not be members of domain groups and hence we need a user/object membership cardinality constraint on those groups. Similarly, users cannot join more than one of the first level groups which requires a mutual exclusion constraint between those groups.

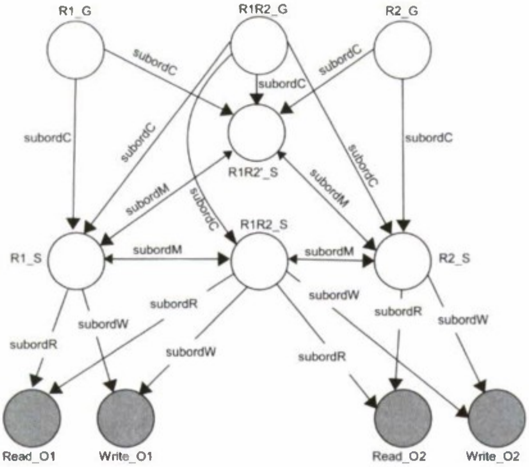


Fig. 8. An equivalent g-SIS^c configuration for the RBAC$_0^{rw}$ model

purpose of our construction, we consider an RBAC$_0$ model with only read and write permissions to objects. Thus for every object, we have two permissions: one to read and the other to write that object. We denote this read-write RBAC$_0$ model as RBAC$_0^{rw}$.

Consider two roles R1 and R2 and two objects O1 and O2. As mentioned earlier, we have two permissions (read and write) for each object, resulting in a total of four permissions. Suppose R1 is assigned permissions to read and write O1 and R2 is assigned permissions to read and write O2. Figure 8 shows an example construction of RBAC$_0^{rw}$ model with two roles R1 and R2 and two objects O1 and O2 in g-SIS^c. The first level of unshaded groups (R1_G, R2_G and R1R2_G) represent groups for user-role assignment. A user may be a member of one of these groups. For instance, users in R1_G have role R1 while users in R1R2_G are assigned to roles R1 and R2. The second level of light-gray groups represent sessions. Note that the group R1R2′_S represents activating a session with no roles assigned. The second level of groups are related to unshaded groups using subordCrelations specifying the rules for subject creation (similar to session in RBAC$_0^{rw}$). Note that subjects may move between the light-gray groups as per the subordMrelation defined. This allows users to activate and de-activate a role dynamically.[4] Finally, the last level of dark-gray groups represent object permissions. Groups Read_O1 and Write_O1 and Read_O2 and Write_O2 represent permissions for objects O1 and O2 respectively. These groups are related to the light-gray groups as per the requirements of permission-role assignment. In the figure, roles R1 and R2 have read and write permissions to objects O1 and O2 respectively. Thus users assigned to both roles R1 and R2 have read and write permissions to both objects O1 and O2. The subordRand subordWrelations defined in figure 8 reflect this configuration.

[4] Again, constraints are necessary for a complete construction. For instance, a subject may move from R1_S or R2_S to R1R2_S only if the user who owns the subject is a member of R1R2_G. Additional constraints are also necessary to ensure that users are not assigned to more than one of R1_G, R2_G and R1R2_G.

4 Conclusion and Discussion

We presented some of design choices for a connected, undifferentiated group g-SIS model and demonstrated its agility with respect to the ease with which changes to information flow/sharing pattern in classic LBAC models can be efficiently handled. We also showed an equivalent representation of an RBAC model with read-write permissions. Because of this result and as per [23], we claim it is feasible to configure Discretionary Access Control policies in g-SIS^c. This positive result allows a system to use the same trusted computing base to configure any of these policies. Prior work on non-transitive information flow in the literature (see [29] for example) is relevant in this context. However, g-SIS is far richer and brings in additional concepts such as subject creation and movement subordination. Furthermore, g-SIS accommodates various useful semantics for group operations such as join and leave for users and add and remove for objects as illustrated in [4,5].

Another area of related work is that of Dynamic Coalition (see for example [30,31]). This problem is concerned about forming a coalition amongst different organizations, for instance, in response to a crisis. Most of the security research in this domain has been carried out in the enforcement or E-layer with the exception of a few. (For instance, in [32,33], the authors focus on enriching role-based access control to address the challenges involved in dynamic coalition.) While dynamic coalition is a very broad and large-scale problem, the focus of g-SIS models is more on information sharing. Specifically, it focusses on read and write permissions to objects and containing subject level information flow. We believe that g-SIS policy models can be beneficially used in dynamic coalition scenarios.

Our future work involves formal specification and analysis of a connected group g-SIS model. In our prior work [4,5,6,34], we have formally specified and analyzed an isolated group g-SIS model. We are exploring candidate core security properties for the connected g-SIS model similar to those of the isolated model. A major challenge in the connected model is that relationships are not static like that of LBAC models. Modern information sharing scenarios are dynamic and inter-group relationships change over time. This complicates information flow analysis in the connected model. For instance, information may flow from group g1 to g3 even if g1 and g3 never existed at the same time (it may currently flow from g1 to g2 and from g2 to g3 in the future). Thus, unlike LBAC, information flow properties tend to be temporal in nature in g-SIS^c.

Acknowledgments

The authors are partially supported by grants from NSF, AFOSR MURI, THECB, State of Texas Emerging Tech. Fund and Intel Corp.

References

1. Saltzer, J., Schroeder, M.: The protection of information in computer systems. Proceedings of IEEE 63(9), 1278–1308 (1975)
2. Wikipedia: Analog hole (September 2009) (Online; accessed December 15, 2009)

3. TCG: TCG specification architecture overview (August 2007), `http://www.trustedcomputinggroup.org`
4. Krishnan, R., Sandhu, R., Niu, J., Winsborough, W.: A conceptual framework for group-centric secure information sharing. ACM Symposium on Information, Computer and Comm. Security (March 2009)
5. Krishnan, R., Sandhu, R., Niu, J., Winsborough, W.H.: Foundations for group-centric secure information sharing models. In: Proc. of ACM Symposium on Access Control Models and Technologies (2009)
6. Krishnan, R., Sandhu, R., Niu, J., Winsborough, W.: Towards a framework for group-centric secure collaboration. In: Proceedings of IEEE International Conference on Collaborative Computing (2009)
7. Krishnan, R., Sandhu, R., Ranganathan, K.: PEI models towards scalable, usable and high-assurance information sharing. In: ACM Symposium on Access Control Models and Technologies (SACMAT 2007), pp. 145–150. ACM, New York (2007)
8. Sandhu, R.: The PEI framework for application-centric security. In: Proceedings of 5th International Conference on Collaborative Computing: Networking, Applications and Worksharing (2009)
9. Sandhu, R., Ranganathan, K., Zhang, X.: Secure information sharing enabled by trusted computing and PEI models. In: Proc. of ACM Symp. on Inf. Computer and Comm. Security, pp. 2–12 (2006)
10. Sandhu, R., Samarati, P.: Access control: Principles and practice 32(9), 40–48 (1994)
11. OrangeBook: Trusted Computer System Evaluation Criteria. DoD National Computer Security Center (December 1985)
12. Graham, G., Denning, P.: Protection-principles and practice. In: Proceedings of the AFIPS Spring Joint Computer Conference, vol. 40, pp. 417–429 (1972)
13. Lampson, B.: Protection. ACM SIGOPS Operating Systems Review 8(1), 18–24 (1974)
14. Graubart, R.: On the Need for a Third Form of Access Control. In: Proceedings of the 12th National Computer Security Conference, pp. 296–304 (1989)
15. McCollum, C., Messing, J., Notargiacomo, L.: Beyond the pale of MAC and DAC - defining new forms of access control. In: Proceedings of the 1990 IEEE Symposium on Security and Privacy, pp. 190–200 (1990)
16. Abrams, M., Heaney, J., King, O., LaPadula, L., Lazear, M., Olson, I.: Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy. In: Nat. Comp. Sec. Conf. (1991)
17. Park, J., Sandhu, R.: Originator control in usage control. In: Policies for Distrib. Syst. and Networks (2002)
18. Bell, D., La Padula, L.: Secure computer systems: Unified exposition and multics interpretation. Technical Report ESD-TR-75-306 (1975)
19. Denning, D.: A Lattice Model of Secure Information Flow. Communications of the ACM 19(5), 236–243 (1976)
20. Sandhu, R.: Lattice-Based Access Control Models. IEEE Computer 26(11), 9–19 (1993)
21. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Trans. on Inf. and Syst. Security (TISSEC) 4(3), 224–274 (2001)
22. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-Based Access Control Models. IEEE Computer, 38–47 (1996)
23. Osborn, S., Sandhu, R., Munawer, Q.: Configuring Role-Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. ACM Trans. on Inf. and Syst. Security 3(2), 85–106 (2000)
24. Park, J., Sandhu, R.: The UCON$_{ABC}$ usage control model. ACM Transactions on Information and System Security (TISSEC) 7(1), 128–174 (2004)

25. XACML: OASIS eXtensible Access Control Markup Language (April 2009), `http://www.oasis-open.org/committees/xacml/`
26. Levin, R., Cohen, E., Corwin, W., Pollack, F., Wulf, W.: Policy/mechanism separation in Hydra. In: 5th ACM Symposium on Operating Systems Principles, pp. 132–140 (1975)
27. Rafaeli, S., Hutchison, D.: A survey of key management for secure group communication. ACM Computing Surveys, 309–329 (September 2003)
28. Badger, L., Sterne, D.F., Sherman, D.L., Walker, K.M., Haghighat, S.A.: Practical domain and type enforcement for unix. In: SP 1995: Proceedings of the 1995 IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 66. IEEE Computer Society, Los Alamitos (1995)
29. Foley, S.N.: A model for secure information flow. IEEE Symposium on Security and Privacy, 248–258 (1989)
30. Phillips Jr., C.E., Ting, T., Demurjian, S.A.: Information sharing and security in dynamic coalitions. In: SACMAT 2002: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, pp. 87–96. ACM, New York (2002)
31. Shands, D., Jacobs, J., Yee, R., Sebes, E.: Secure virtual enclaves: Supporting coalition use of distributed application technologies. ACM Transactions on Information and System Security (TISSEC) 4(2), 103–133 (2001)
32. Freudenthal, E., Pesin, T., Port, L., Keenan, E., Karamcheti, V.: drbac: Distributed role-based access control for dynamic coalition environments. In: ICDCS 2002: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS2002), Washington, DC, USA, pp. 411–420. IEEE Computer Society, Los Alamitos (2002)
33. Cohen, E., Thomas, R.K., Winsborough, W., Shands, D.: Models for coalition-based access control (CBAC). In: SACMAT 2002: Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, pp. 97–106. ACM, New York (2002)
34. Krishnan, R., Niu, J., Sandhu, R., Winsborough, W.: Stale-safe security properties for group-based secure information sharing. In: Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering, pp. 53–62. ACM, New York (2008)

A Predictive Model for Cache-Based Side Channels in Multicore and Multithreaded Microprocessors

Leonid Domnitser, Nael Abu-Ghazaleh, and Dmitry Ponomarev

Computer Science Department
State University of New York at Binghamton
Binghamton, NY 13902
{lenny,nael,dima}@cs.binghamton.edu

Abstract. A side channel is an information channel that unintentionally communicates information about a program as a side effect of the implementation. Recent studies have illustrated the use of shared caches as side channels to extract private keys from computationally secure cryptographic applications. The cache side channel is imperfect in the sense that the attacker's ability to detect cache leakage of critical data is limited by the timing issues. Moreover, some detected leakages are due to non-critical data. Thus, it is difficult to assess the degree of vulnerability given the imperfect nature of the side-channel. Similarly, when solutions that further degrade the quality of the channel, but do not necessarily close it completely, are employed, it is difficult to evaluate their effectiveness. To address this need, this paper proposes a mathematical model to evaluate the expected leakage in a cache as a function of the cache parameters and the victim application behavior. We use simulation to quantify these parameters for typical attack scenarios to validate the model. We demonstrate that the proposed model accurately estimates side channel leakage for for AES and Blowfish encryption and decryption on a variety of cache configurations.

Keywords: architecture, security, side channel attack, caches.

1 Introduction

In reeent years, security has emerged as one of the key design issues in computing and communication systems. Security solutions typically rely on a set of cryptographic algorithms, sueh as symmetric ciphers, publie-key eiphers, and hash functions. The strength of modern cryptography makes it infeasible for the attackers to uncover the seeret keys used in these algorithms by brute-force trials, differential [9] or linear cryptanalysis [14]. Instead, almost all known attaeks on the secret keys today exploit weaknesses in the physieal implementation of the system performing the encryption, rather than exploiting the mathematieal properties of the eryptographie algorithm itself.

A subtle form of vulnerability in the physical implementation of otherwise secure systems is a possible leakage of information through unintended (or side) ehannels. The leaked information is called *side-channel information*, and the attacks exploiting side-channel information leakage are called *side-channel attacks* [1,19,21]. Examples of side-channels include observation of execution time, power consumption, heat, electromagnetic radiation, or even sound emanating from a deviee [21]. A large number

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 70–85, 2010.

of side-channel attacks have been successfully demonstrated against a range of software and hardware security mechanisms: they have been used to break many cryptosystems including block ciphers (such as DES, AES, Camellia, IDEA, and Misty1), stream ciphers (such as RC4, RC6, A5/1, and SOBER-t32), public key ciphers (such as RSA-type ciphers, ElGamal-type ciphers, ECC, and XTR), signature schemes, message authentication code schemes, cryptographic protocols, and even the networking subsystems[21]. Thus, it is critical to build systems that are immune to side-channel attacks.

Traditionally, side-channel attacks were used to break simple systems such as smart cards. However, a new class of attacks that exploit the shared caches in microprocessors as side-channels had recently emerged as a serious security threat [26,27,12,24,18,19,1,3,5]. The nature of this new threat is rooted in the ability of modern microprocessors to execute several programs concurrently on the same chip to exploit so-called Thread-Level Parallelism (TLP). TLP is exploited by the processor designers in two ways: Simultaneous Multithreading (SMT) and Chip Multiprocessing (CMP, also called multicore). In an SMT processor [25], several independent programs are simultaneously executing on the same processing core, and most of the core's resources, including the on-chip caches, are shared among the threads. In contrast, CMPs consist of completely replicated processing cores that share only a small subset of resources such as lower level caches, main memory and I/O pins.

Consider the concurrent execution of two programs in an SMT environment – a security-critical encryption kernel (which we refer to as the "victim") and a process performing an attack on the secret key used by the victim (which we refer to as the "attacker"). By sharing the data cache with the victim, the attacker can detect the victim's cache accesses when the victim evicts the data belonging to the attacker. The detection is possible because on its next access to the same data, the attacker will miss into the cache, and cache misses can be easily distinguished from hits by using timing instructions readily available in most modern innstruction sets. Several studies [19,1,3,24] showed that the information leaked through the cache side channel is often sufficient to reconstruct the full secret key in a short period of time. Similar leakage is possible through the shared lower-level caches in multicore system, even without multithreading in individual cores.

Although cache-based side-channel attacks have been demonstrated, a successful attack involves gleaning of the critical information from an imperfect channel. In particular, some memory accesses may not be leaked at all (we explain the reasons for that in detail in the later sections). Moreover, some non-critical accesses may be detected; these accesses do not correlate with the secret key and therefore add noise to the information collected from the side-channel. Thus, key reconstruction involves a significant effort, depending on the amount and quality of the information detected from the side-channel. If the amount of useful information collected through the side channel is small, key reconstruction may require prohibitive computational overhead, or just fail.

Being able to quantify this relationship between a side channel leakage properties and the computational difficulty of compromise is critical. It allows quantitative evidence of vulnerabilities. Moreover, it may be impossible or extremely expensive to completely shut down the side-channel[26,27]. Thus, low complexity solutions that reduce the quality of the channel may be of interest. However, it is difficult to accept such defenses based on informal evidence; being able to formally quantify the security of imperfect channel can lead to effective solutions that have acceptable complexity while providing sufficient security.

To help the computer designers implement the right level of protection against cache-based attacks, two key questions have to be addressed: 1) How much information is leaked through the side channel, and 2) what is the effort required to convert this information into a full secret encryption key. In this paper, we address the first question and develop a simple analytical model to predict the amount of critical information leakage through the cache-based side channel. The model takes into account the capabilities of the attacker, the parameters of the victim process and the hardware configuration of the cache. We validate the developed model through cycle-accurate simulations of two encryption kernels (AES and Blowfish) and demonstrate that the side channel leakege predicted by the analytical model matches simulation-based results. Finally, we explain how the designers can use the proposed model to reason about the threat level, the impact of different attack optimizations, and the impact of possible defense approaches.

The remainder of the paper is organized as follows. We review the AES and Blowfish algorithms and analyze why they are vulnerable to cache-based attacks in Section 2. Scsction 3 describes the proposed leakage prediction model. In Section 4, we present the simulation methodology and simulation results to validate the model. Section 5 reviews the related work and we conclude in Section 6.

2 Background

In this section we describe how the Advanced Encryption Standard (AES) and Blowfish encryption lend themselves to exploitation by side channel attackers. We also explain how a side channel attack through the L1 data cache works. An attack on last level cache in a multicore system can be performed in a similar fashion.

2.1 The Advanced Encryption Standard (AES)

AES, the Advanced Encryption Standard, is a widely used symmetric block cipher. It encrypts and decrypts 128-bit data blocks using either a 128-, 192-, or 256-bit key. Each block is encrypted in 10 rounds of mathematical transformations. To achieve high performance, AES implementations use precomputed lookup tables instead of computing the entire transformation during each round. The indexes to these tables are partially derived from the secret key, thus by detecting the cache sets accessed by the victim (through the side channel observations), the attacker can derive some information about parts of the secret key. By using multiple measurements, the entire key can be successfully reconstructed. The version of the AES code that we use in this study [6] employs five tables (1KB each) for both encryption and decryption. The first four tables are used in the first nine rounds of encryption/decryption, and the fifth table is used during the last round. Separate sets of tables are used for encryption and decryption. More details on the AES encryption algorithm and specific side channel attacks on AES can be found in [24].

2.2 The Blowfish Encryption Algorithm

Blowfish [2] is a keyed, symmetric block cipher, included in a large number of cipher suites and encryption products. Blowfish has a 64-bit block size and a variable key length from 32 up to 448 bits. It is a 16-round Feistel cipher and uses large key-dependent S-boxes. The algorithm keeps two subkey arrays: the 18-entry P-array and four 256-entry S-boxes. The S-boxes accept 8-bit input and produce 32-bit output. One entry of the P-array is used every round, and after the final round, each half of the data block is XORed with one of the two remaining unused P-entries. The F-function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo 2^{32} and XORed to produce the final 32-bit output. Again, just in the case with AES, the accesses to S-boxes are the critical accesses that can reveal the key-related information through the cache side channel.

2.3 Cache-Based Side Channel

A dangerous side channel exists if an attacker can determine which table rows, which we call *critical data*, are accessed. A shared memory cache can carry such a side channel. The time to access data present in the cache (a cache hit) is different from the time to access data not in the cache (a cache miss), so it is possible to tell which data is in the cache by measuring access time.

Caches typically have a set-associative organization. Each memory location is part of a multi-byte data block called a cache line, and several lines are grouped into a cache set. When data is loaded, an entire line is brought in and is deterministically mapped, by address, into a specific set. Multiple lines coexist in a set (the number of lines in a set is the associativity of the cache), but when new lines are loaded, an old line must usually be evicted. This is done according to a replacement policy, such as evicting the least recently used (LRU) line.

When a line of critical data is loaded, it replaces some other line. If the cache is shared with an attacker, the evicted line may belong to the attacker. By later accessing that line, and timing that access, the attacker learns whether the victim accessed an address mapping to the same set.

The Attack. Given such a side channel, an attack is relatively straightforward. An attacker fills the entire cache, ensuring that any memory access by the victim will evict the attacker's data. After a cryptographic operation, the attacker returns to each cache set and accesses the same data to determine if it misses, indicating that the victim accessed the set.

The attacker needs to perform the following steps:

- Gain user-level access to the computer performing cryptography.

- It must also be aware of the cache configuration—number of sets, associativity, line size, replacement policy, etc. This might be known ahead of time, or it might be programatically deduced.
- For a side channel to be useful, the attacker must be somehow synchronized with the victim. For simplicity, we assume that a synchronous attack is possible. That is, the attacker can trigger the cryptographic process, say, by an inter-process communication or networking mechanism. If an attacker can trigger single–data-block encryption or decryption, it need not worry about keeping pace with the victim to ensure that the cache stays full of attacker data.
- The attacker cannot "look within" a line. When an attacker line is evicted, all it can learn is that the victim accessed a location within a particular line, not the specific critical data index within the line. This is the nature of the side channel—an attacker will have to do some brute force work.
- The critical data accessed can be difficult to determine, not just because of line-size granularity, but because other, unrelated victim data can map to the same set, creating noise in the side channel.

3 Model for Side Channel Leakage Prediction

The goal of the proposed model is to predict the probability that a critical cache access (that is, the access to the critical data that is dependent on the secret key) is exposed on a cache-based side channel, both in aggregate and per each cache set. More precisely, we predict the conditional probability that the attacker detects a memory access on the side channel, given that there was access to critical data. This probability can be expressed as $P(D|C)$, where D is the event of detection, and C is the event of a critical access. $P(D|C)$ is defined to be equal to $\frac{P(D\cap C)}{P(C)}$. Using a few basic algebraic transformations of this definition, we obtain a simple statement of Bayes' theorem, which gives the relationship between a conditional probability and its inverse:

$$P(D|C) = \frac{P(C|D)\,P(D)}{P(C)} \tag{1}$$

Table 1. Summary of symbols

A	event of an access
α	number of memory accesses
C	event of critical access
D	event of detected access
m	number of lines used in a set
N	number of sets
s	cache set number
T_a	time between repeat accesses by attacker
T_v	time between repeat accesses by victim
w	cache associativity

This formula is the basis of the proposed model. We predict the variables $P(C|D)$ (the conditional probability that there was a critical access, given that the attacker detected an access), $P(D)$, and $P(C)$ based on properties of the attacker and victim programs, and the system on which they execute. In the next section, we measure the variables through simulation and evaluate our predictions.

$P(D)$ and $P(C)$ refer to the probabilities of events D and C *when an access occurs*, not for any instruction. That is, we assume an initial condition A, that there is a memory access. So $P(D)$ is a shortened notation for $P(D|A)$. Accordingly, $P(D)$ and $P(C)$ are affected by the real timeline of CPU cycles, but are calculated relative to the timeline of memory access instructions only.

The events A, C, and D can refer to all accesses, or can be restricted accesses to a subset of data. We consider both aggregate measures and those restricted to individual cache sets. In the latter case, A_s, C_s, and D_s refer to the events of accesses to set s.

The rest of this section analyzes the components variables of the above Bayesian formula, then combines the variable predictions into a single formula to predict side channel exposure of critical data.

3.1 Estimating Critical Accesses ($P(C)$)

We define the probability of a critical access, $P(C)$ as the average rate of critical accesses. This probability defines how many accesses during the execution of a cryptographic program are critical, if the total number of accesses is α. This value is an invariant property of the implementation of a cryptographic algorithm, and can be estimated through static analysis or profiling.

In our Bayesian prediction formula, $\frac{1}{P(C)}$ is a constant for a given program. However, it varies among cryptographic algorithms (and implementations), and between encryption and decryption routines.

3.2 Access Detection ($P(D)$)

The probability that the attacker detects an access, $P(D)$, is the number of detected accesses out of the total number of accesses.

$$P(D) = \frac{\alpha_D}{\alpha} \tag{2}$$

$P(D)$ is 100% for a perfect attacker, which measures the victim without error after every instruction. Realistically, there are several reasons that a memory access may be hidden from the side channel. The number of hidden accesses is $\alpha_{\neg D}$, and $P(D)$ can be restated in terms of hidden accesses:

$$P(D) = 1 - \frac{\alpha_{\neg D}}{\alpha} \tag{3}$$

The simplest reason that a memory access may not be detected is a cache hit. The attacker only sees a victim's access if it misses into the cache and evicts attacker's data. The attacker's data is not automatically placed back in the cache when the victim evicts it; rather, the attacker scans and refills the particular cache location at some point after the victim accesses it. In the time it takes the attacker to traverse the cache, the victim may have performed several accesses to the same cache location, and all but one of those will remain undetected by the attacker.

The percentage of accesses hidden from the side channel by the cache hits depends on the victim's pattern of access to individual critical data entries and the speed with which an attacker returns to each set. These behaviors can vary depending on victim and attacker implementations, as well as the cache configuration.

For a fixed cache configuration, the victim's "rate of return" to a particular critical data entry depends on the input data and the secret key. The average rate is a property of the cryptographic code. Since the side channel operates at the granularity of cache lines, the cache line size also affects the hit rate. The victim does not necessarily have to access the exact same critical data to hit into the cache, but must access data within a resident cache line. Larger cache lines, besides confounding multiple possible critical data addresses when an access *is* detected, increase the likelihood of cache hits which are hidden from the side channel.

The attacker can also be accelerated by larger cache lines, since fewer memory accesses are required to scan the entire cache. If the rate of return for both the victim and attacker scale linearly with the line size, then the effect is canceled out. However, a victim automatically exploits the increased hit rate, while an attacker may have to address synchronization issues that result. In a synchronous attack that is synchronized at block encryption boundaries, increased line size helps only the victim. Even in a purely asynchronous attack, where "synchronization" is done by post-attack analysis, the attacker must still emit or save its timing results after each cache traversal, which will happen more often with shorter traversal times.

Depending on the attack code, the cache dimensions—number of sets and associativity—can affect the speed of the attack. At the cost of higher code complexity, an attacker can reduce the number of accesses it must perform by accessing each set just once[1]. If this sort of an attack is used, then the attacker can traverse a highly associative cache with fewer sets faster than it can traverse a less associative cache (with more sets) of the same size. Similar to increasing line size, increasing associativity can increase the attack speed and decrease hidden critical accesses, but only if this optimized attack is used, and if synchronization issues are handled.

Memory accesses can also be hidden from the side channel by design. Hardware or software defense mechanisms can reduce $P(D)$. A perfect defense mechanism reduces $P(D|C)$ to 0. Of course, if no critical accesses are detected, the side channel does not exist. Specific defense mechanisms are outside the scope of this paper, but they can be captured in this portion of the model.

3.3 Estimating Critical Accesses Given Detection ($P(C|D)$)

$P(C|D)$ is the fraction of detected accesses that are critical. This represents how clean the signal on the side channel is—100% means that every access that is detected is useful to the attacker. A real side channel, however, has sources of noise. Noise is expressed

[1] Assuming the cache replacement policy is Least Recently Used (LRU), the attacker can ensure that it always accesses the LRU line of a set. If that is a hit, the other lines in the set will also hit.

as $P(\neg C|D)$, or the probability that there is no critical access, given a detected access. (With our assumptions, C also means that there was a non-critical access.) Since we consider the cause of noise, not of signal, we represent $P(C|D)$ as $1 - P(\neg C|D)$.

Even with no complicating factors, some noise is inherent in the cryptographic implementation. A victim's access to non-critical data that maps to the same cache set as critical data cannot be distinguished from a critical access. The attacker, in its analysis stage, can try to filter out a pattern of noise. An attack must have tolerance for noise, so that some brute force trials can be performed while still utilizing data from the side channel as a hint.

Noise can also come from the attacker's side, in the form of instruction cache misses, from misses in the translation lookaside buffer, from the operating system, or from anything else that may confound timing results.

3.4 Model Formalization

We now unify the ideas presented above into a formal predictive model. We return to our Bayesian formula, predicting side channel exposure in a single set:

$$P(D_s|C_s) = \frac{P(C_s|D_s)P(D_s)}{P(C_s)} \tag{4}$$

The probability of a critical access is simply the ratio of critical accesses to total accesses:

$$P(C_s) = \frac{\alpha_{C,s}}{\alpha_s} \tag{5}$$

To model the probability of a detected access, we first consider the average likelihood that an access will be hidden by hitting into the cache. Repeated victim's access to a cache location before the attacker scans and refills that location will be hidden from the side channel. Consequently, in the time it takes an attacker to return to a location, T_a, all accesses after the first one will be hidden. The time between repeated accesses to that location is T_v. The probability of an access to a single location being detected, then, is $\frac{T_v}{T_a}$, assuming there is an intervening victim access to the location ($T_v \leq T_a$). T_v, on average across accesses during an attacker traversal, is expressed as $\frac{T_a}{\alpha_{location}}$. So we obtain:

$$P(D_{location}) = \frac{1}{\alpha_{location}} \tag{6}$$

Since we assume that a cache access has taken place, $\alpha > 0$, this value is defined.

Next, we expand our model to a cache set, rather than a single location. There is some number of lines of data mapping to set s, which we call m_s. The cache is w-way associative. If $m_s \leq w$, then all the data fits in the set without eviction, and accesses to all these data can be detected. In this case $P(D_s) = \frac{m_s}{\alpha_s}$. When $m_s > w$, only $\frac{w}{m_s}$ of the data lines used are resident in the cache, and can possibly leak information. This is a "fit factor" that limits the probability of detection in a set. Generalizing, we obtain:

$$P(D_s) = \frac{m_s \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_s} \tag{7}$$

$m_s \cdot \min\left(\frac{w}{m_s}, 1\right)$ can be thought of as the "pressure" on the cache set.

We now model the probability of critical access, given detected access, which can be expressed as $1 - noise$. The simple noise we consider is caused by non-critical accesses. This model considers the number of critical accesses to a set $\alpha_{C,s}$, which along with non-critical accesses $\alpha_{\neg C,s}$, make up all accesses α_s. We start with the definition of conditional probability:

$$P(C_s|D_s) = \frac{P(C_s \cap D_s)}{P(D_s)} \tag{8}$$

$P(D_s)$ is already modeled. The model for $P(C_s \cap D_s)$ is very similar—we just look at the subset of detected accesses that are critical. $m_{C,s} \leq m_s$ is the number of critical data lines mapping to set s. We restrict $P(D)$ to just these lines:

$$P(C_s \cap D_s) = \frac{m_{C,s} \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_s} \tag{9}$$

Thus, the equation for $1 - noise$ is the following:

$$P(C_s|D_s) = \frac{\left(\frac{m_{C,s} \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_s}\right)}{\left(\frac{m_s \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_s}\right)} = \frac{m_{C,s}}{m_s} \tag{10}$$

We can now unify the model into a single formula:

$$P(D_s|C_s) = \frac{\left(\frac{m_{C,s}}{m_s}\right)\left(\frac{m_s \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_s}\right)}{\left(\frac{\alpha_{C,s}}{\alpha_s}\right)} \tag{11}$$

Our single-set model simplifies to:

$$P(D_s|C_s) = \frac{m_{C,s} \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_{C,s}} \tag{12}$$

Finally, we consider the entire cache rather than a single set. We ignore sets without critical accesses, since the condition of the probability is that there were critical accesses, and take the average:

$$P(D|C) = \frac{\sum_{s=0}^{N} \begin{cases} 0 & : \alpha_{C,s} = 0 \\ \frac{m_{C,s} \cdot \min\left(\frac{w}{m_s}, 1\right)}{\alpha_{C,s}} & : otherwise \end{cases}}{\sum_{s=0}^{N} \begin{cases} 0 : \alpha_{C,s} = 0 \\ 1 : otherwise \end{cases}} \tag{13}$$

This model predicts the effectiveness of a side channel using only a simple profile of the victim (distribution of critical and overall cache accesses within a period of attack), and cache associativity.

4 Model Validation Methodology

To validate the model proposed in the previous section, we performed cycle-accurate simulations of AES and Blowfish encryption algorithms running alongside the idealized attacker on an SMT processor. We used M-Sim 3.0 [17]—a SMT and CMP simulator that was derived from Simplescalar 3.0d [7]. The crypto programs were compiled on an Alpha AXP machine running Tru64 UNIX, using the native C compiler with `-O4 -fast -non_shared` optimization flags. We simulated an 8-way processor with 128-entry Reorder Buffer and a 32KB L1 data cache under several configurations.

We simulated both an 8-way set-associative cache, and a direct-mapped (1-way) cache. The 8-way cache is a typical configuration, while the direct-mapped cache was used to exercise the fit factor of the model. Both caches used 32 byte lines, so the caches had 128 and 1024 sets, for 8-way and direct-mapped, respectively.

We simulated the execution of an idealized attacker as a separate thread alongside an encryption or decryption process. It is "idealized" because it is implemented with simulator support to synchronize perfectly with cryptographic block operations. The only noise, therefore, occurred during the core cryptographic operation, and the attacker did not operate under time constraints. This is the best an attacker can do, and a is worst-case bound on security.

We ran simulations of 1,000 truly random [20] input data blocks. The experiments included all permutations of the following operations:

- Algorithm: AES or Blowfish
- Operation: encryption or decryption
- Cache configuration: 8-way or direct-mapped

Our simulations outputted memory access traces; a script used these data to generate predictions. We also generated attacker traces which labeled memory accesses as detected or not. For each block operation (that is, each instance of the attack), the model matches the measured detection rates. This makes sense, because the same data block is used both to profile the victim for the model and to measure the side channel. We also show how the average side channel that we measure as a profile is a good predictor for individual blocks. Our experiments show low variation in side channel leakage from block to block.

5 Results and Evaluation

Figure 1 shows the aggregate detection rate averaged across 1,000 cryptographic block operations. This is the measured rate of detection, which matches the rate predicted by our model for each of the same blocks. The average value of $P(D|C)$ is 76% for AES and 87% for Blowfish on the 8-way cache. The direct-mapped cache exhibits similar results, 77% and 88% for AES and Blowfish, respectively. Figure 2 shows same side channel leakage broken down by set. The 8-way cache has a fairly consistent distribution of accesses among its 128 sets, since the critical data tables are spread evenly across all sets. The direct-mapped cache only shows leakage in 256 sets for AES and 268 sets for Blowfish, out of 1024 available sets. This is because only those sets have critical data (since direct mapped cache has a larger number of sets).

Figure 3 shows, for each experiment, how the detection rate for each block compares to the average rate across all the blocks. These graphs show bell curves in a small range around the average, with a standard deviation of 2% in all cases. The maximum deviation is 7% for AES and 8% for Blowfish. Therefore, our simple predictive model which is based on the average detection rate across all blocks predicts the detection within individual blocks with an error of only 8% in the worst case.

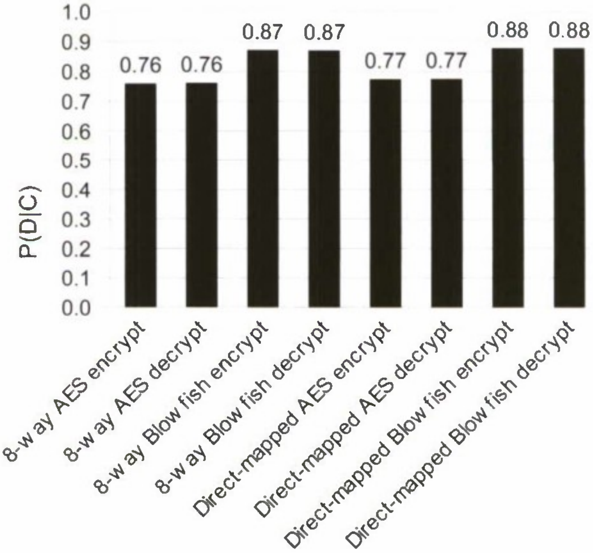
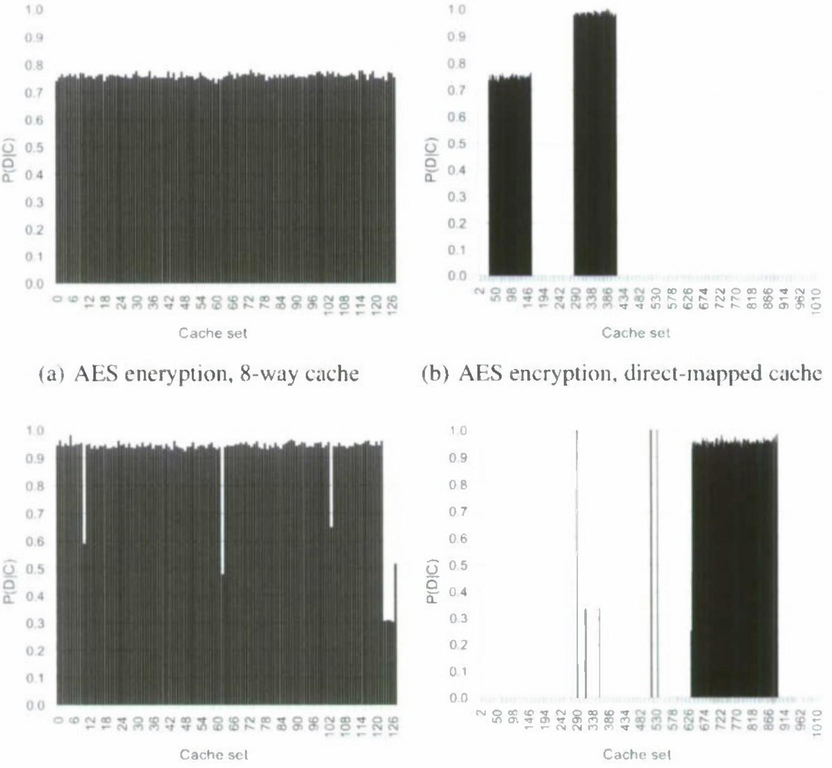


Fig. 1. Aggregate detection rate. This graph shows the detection rate ($P(D|C)$) as an average across 1,000–data block cryptographic operations. Predicted and measured detection rates are equal for each block operation.

Therefore, our experiments confirm that the results obtained by the proposed model closely match the simulated results. Furthermore, they demonstrate that different input data causes only a low level of variation in side channel leakage, so simple metrics, such as estimated average characteristics across all blocks can be used for accurately predicting the amount of leaked data on a block-by-block basis.



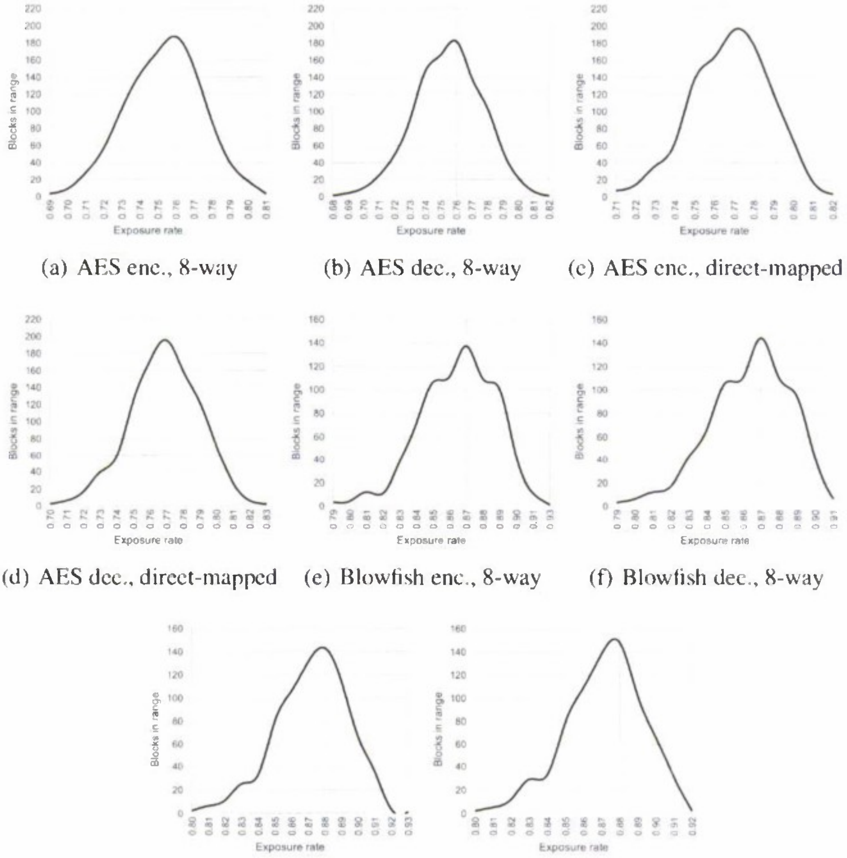
(a) AES encryption, 8-way cache

(b) AES encryption, direct-mapped cache

(c) Blowfish encryption, 8-way cache

(d) Blowfish encryption, direct-mapped cache

Fig. 2. Detection by set. These graphs show detection rate in each set ($P(D_s|C_s)$) as averages across 1,000–data block encryptions (decryption graphs are omitted because they are nearly identical). Predicted and measured detection rates are equal for each block operation.



(a) AES enc., 8-way (b) AES dec., 8-way (c) AES enc., direct-mapped

(d) AES dec., direct-mapped (e) Blowfish enc., 8-way (f) Blowfish dec., 8-way

(g) Blowfish enc., direct-mapped (h) Blowfish dec., direct-mapped

Fig. 3. Each graph shows the number of block operations with side channel leakage within a given 1% range

6 Related Work

The security of cryptographic implementations with respect to side-channel attacks has not been widely investigated. Despite the presence of a number of a solutions to side-channel problems that do not perfectly close the channel [10,15], the security properties of side-channels and the effectiveness of such imperfect solutions were open questions. Micali and Reyzin were the first to present a theoretical analysis of general side-channel attacks [16]. Using very general assumptions, this model defines the notion of an abstract computer and a leakage function that together can capture almost all instances of side channels. However, the overly general assumptions make it difficult to apply this analysis to particular algorithms (e.g., DES or AES) or for specific side-channels.

Standaert *et al.* started from Micali and Reyzin model and specialized it for more practical situations [23]. Specifically, they restricted some of the assumptions to a range that corresponds to relevant adversary and leakage models. Moreover, they show how to map the abstract computational model to physical instances such as circuits and operations. Although this model brings the original model by Micali and Reyzin closer to practice, it models the leakage and adversary abstractly using information theoretic principles. More recently, Standaert *et al.* created a uniform model of side-channel attacks to address the problem of how compare different algorithm implementations and defense mechanisms in a way that enables comparing them [22].

Kopf and Basin developed an information theoretic model of side-channels [13]. Like our model, they restrict their analysis to the amount of information leaked from the channel. This model considers a generic side channel, and does not capture the detailed operation of cache based side-channel attacks that we characterize in this paper.

Both software and hardware solutions to address cache-based side channel attacks have been proposed. On the software side, the main idea is to rewrite the code of the encryption algorithms such that known side channel attacks are not successful. Examples of such techniques include avoiding the use of table lookups in AES implementations, preloading the AES tables into the cache before the algorithm starts, or changing the table access patterns [18,24,4,21]. The limitation of the software solutions is that that they are tied up to a specific algorithm/attack, do not provide protection in all cases, are subject to errors on the part of programmers, and often result in significant performance degradation [26]. Another recent approach to address side channel attack is by dedicating special functional units and ISA instructions to support a particular cryptographic algorithm. An example of this approach is the Intel AES instruction [11]. This, however, requires non-trivial hardware and software changes and only protects against the attacks on the crypto algorithms that are supported—support has to be re-implemented to defend new algorithms.

In response to the limitations of software solutions, several hardware schemes have been recently introduced. The advantage of hardware solutions is that they prevent the attacks in principle, by eliminating the side channel. The main challenge in these schemes is to keep the impact on the design complexity, cache access time, and performance overhead to the minimum. Following this line of research, a partitioned cache was proposed [8], along with ISA changes to make the cache a visible part of the architecture. Specifically, new instructions are added to define a partition and specify its size and parameters. This scheme requires changes to both the ISA and the cache hardware design and can lead to significant performance degradation. Several alternative cache designs for thwarting cache-based attacks have been proposed by Wang and Lee [26,27]. Partition-Locked Cache (PL cache) design [26] uses cache line locking to prevent evictions of cache lines containing critical data, thus closing the side channel. The main drawback is the performance hit due to cache underutilization, as the locked lines cannot be used by other processes, even after they are no longer needed by the process that owns them. In addition, the PLcache requires system support to control which cache lines should be locked. This support is in the form of new ISA instructions, or OS modifications for marking the regions of memory that contain the AES or RSA tables as lockable. In either case, ISA/compiler/OS modifications are also needed in addition to the hardware changes.

7 Conclusion

Cache-based software side-channel attacks represent a new and serious security threat that exploits parallel processing capabilities of modern processor chips. Defense mechanisms that provide a complete closing of the side channel are expensive and often incur significant performance overhead. A possible alternative is to consider solutions that do not result in a complete elimination of the side channel, but rather attempt to reduce its strength to the levels that make the remaining post-attack effort for the secret key reconstruction infeasible.

To assist system designers with such solutions, we developed an analytical model for estimating the percentage of accesses to the critical data that would be leaked through the cache side channel as a function of the victim's characteristics and the configuration of the cache hardware. We validated the proposed model using cycle-accurate simulation of side-channel attack on two popular encryption kernels (AES and Blowfish) and also described how the model can be used in exploring the design space of low-complexity solutions for cache-based attacks.

Acknowledgements. This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8750-09-1-0137. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies and endorsements, either expressed or implied, of Air Forse Research Laboratory or the U.S. Government.

References

1. Bernstein, D.: Cache-timing attacks on aes (2005), `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`
2. The blowfish encryption algorithm (2009), `http://www.schneier.com/blowfish.html`
3. Bonneau, J., Mironov, I.: Cache-collision timing attacks against aes. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 201–215. Springer, Heidelberg (2006)

4. Brickell, E., Graunke, G., Neve, M., Seifert, J.: Software mitigation to hedge aes against cache-based software side channel vulnerabilities. In: IACR ePrint Archive, Report 2006/052 (2006)
5. Canteaut, A., Lauradoux, C., Seznec, A.: Understanding cache attacks. INRIA Technical Report (2006), `ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-5881.pdf`
6. Daemen, J., Rijmen, V.: The design of rijndael: Aes - the advanced encryption standard. Springer, Heidelberg (2002)
7. Burger, D., Austin, T.: The simplescalar toolset: Version 2.0 (June 1997)
8. Page, D.: Partitioned cache architecture as a side-channel defense mechanism. In: Cryptography ePrint Archive (2005)
9. Biham, E., Shamir, A.: Packaging of multi-core microprocessors: Tradeoffs and potential solutions. Journal of Cryptology 4(1), 3–72 (1991)
10. Goubin, L., Patarin, J.: DES and differential power analysis. In: Proc. of CHES (1999)
11. Gueron, S.: Advanced encryption standard (aes) instruction set (2008)
12. Kong, J., Aciicmez, O., Seifert, J., Zhou, H.: Hardware-software integrated approaches to defend against software cache-based side channel attacks. In: International Symposium on High Performance Computer Architecture (HPCA) (February 2009)
13. Kopf, B., Basin, D.: An information-theoretic model for adaptive side-channel attacks. In: ACM Conference on Computer and Communication Security (CCS), pp. 286–296 (2007)
14. Matsui, M.: Linear cryptanalysis method for des cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
15. May, D., Muller, H., Smart, N.: Randomized register renaming to foil DPA. In: Proc. of CHES (2001)
16. Micali, S., Reyzin, L.: Physically observable cryptography. In: Proc. of Theory of Cryptography Conference (2004)
17. M-sim version 3.0, code and documentation (2005), `http://www.cs.binghamton.edu/~msim`
18. Osvik, D., Shamir, A., Tromer, E.: Cache attacks and countermeasures: the case of aes. In: Cryptology ePrint Archive, Report 2005/271 (2005)
19. Percival, C.: Cache missing for fun and profit (2005), `http://www.daemonology.net/papers/htt.pdf`
20. Random.org (2009), `http://www.random.org/`
21. Side channel attacks database (2009), `http://www.sidechannelattacks.com`
22. Standaert, F.X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Advances in Cryptography, Eurocrypt (2009)
23. Standaert, F.X., Peeters, E., Archambeau, C., Quisquater, J.J.: Towards security limits in side-channel attacks. In: Proc. CHES Workshop (2006)
24. Tromer, E., Shamir, A., Osvik, D.: Efficient cache attacks on aes, and countermeasures. Journal of Cryptology (2009)
25. Tullsen, D., Eggers, S., Levy, H.: Simultaneous multithreading: Maximizing on-chip parallelism. In: International Symposium on Computer Architecture (1995)
26. Wang, Z., Lee, R.: New cache designs for thwarting software cache-based side channel attacks. In: Proc. International Symposium on Computer Architecture (ISCA) (June 2007)
27. Wang, Z., Lee, R.: A novel cache architecture with enhanced performance and security. In: Proc. International Symposium on Microarchitecture (MICRO) (December 2008)

Attack and Defense Modeling with BDMP

Ludovic Piètre-Cambacédès[1,2] and Marc Bouissou[1,3]

[1] Electricité de France R&D, 1 avenue du Général de Gaulle, 92141 Clamart, France
[2] Institut Telecom, Telecom ParisTech, 46 rue Barrault, 75013 Paris, France
[3] Ecole Centrale Paris, Grande Voie des Vignes, 92295 Châtenay-Malabry, France
{ludovic.pietre-cambacedes,marc.bouissou}@edf.fr

Abstract. The BDMP (Boolean logic Driven Markov Processes) modeling formalism has recently been adapted from reliability engineering to security modeling. It constitutes an attractive trade-off in terms of readability, modeling power, scalability and quantification capabilities. This paper develops and completes the theoretical foundations of such an adaptation and presents new developments on defensive aspects. In particular, detection and reaction modeling are fully integrated in an augmented theoretical framework. Different use-cases and quantification examples illustrate the relevance of the overall approach.

Keywords: Security modeling, attack trees, BDMP, risk analysis.

1 Introduction

Graphical attack formalisms are commonly used in security analysis to share standpoints between analysts, enhance their coverage in terms of scenarios, and help ordering them and the related system vulnerabilities by various quantifications. The authors have recently introduced a new approach based on BDMP (Boolean logic Driven Markov Processes) [3], adapting this formalism used in reliability engineering to attack modeling [16]. BDMP have proven to be an original and advantageous trade-off between readability, modeling power, scalability and quantification capabilities in their original domain [2]. The same advantages are expected from their adaptation to the security area. In this paper, we consolidate the theoretical foundations of such an adaptation, and extend it to take into account detection and reaction aspects in an integrated approach. Section 2 presents the state of the art in graphical attack modeling. Section 3 develops, on a theoretical and practical point of view, how BDMP can be changed to model attack scenarios. Section 4 focuses on defensive aspects, presenting the extension developed for detection and reaction modeling. Section 5 presents on-going and future work related to this new approach.

2 State of the Art

The clear interest of the computer security community for graphical attack modeling techniques has led to numerous proposals; they can be grouped into two categories, each being dominated by a specific model:

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 86–101, 2010.

- *Static models*: also called structural models, they provide a global view of the attack, without being able to capture its evolution in time. The dominant type of model is the Boolean-logical tree based approach. Generally known as Attack Trees [21,10], they are present in the literature under different variations: threat trees [1], vulnerability trees [14] etc.
- *Dynamic models*: also called behavioral models, they take into account dependance aspects such as sequences or reactions. Richer than static models, they can be built by hand only in very simple cases. There are two approaches in the other cases:
  - The first one is based on detailed state-graphs capturing the possible evolutions of an attack, automatically generated from formal specifications. Such approaches, initiated by Sheyner et al. with Attack Graphs [22] and followed by other relevant approaches (e.g. [8,7]), are not graphical models per se as they are not directly designed to be graphically manipulated by analysts.
  - The second relies on compact and high-level graphical formalisms, designed to efficiently represent dynamic aspects like sequences or reactions, and to be directly usable by human analysts. In this category, Petri net-based approaches are the most widely known. Attack Nets, one of the first proposals in the domain [11], or PE Nets, a more recent approach with a complete software support [18], are two good representatives.

Each approach allows for a different balance in terms of modeling power, readability, scalability and quantification capabilities. Static models are usually very readable but are lacking in their modeling power and quantification capabilities. Dynamics models are more interesting for these aspects, but often have their own limits in terms of clarity and scalability. Note that these statements are also relevant in the domain of reliability and safety modeling [12,17], where similar approaches have been historically first used, modeling system component failures instead of attacker actions and security events.

3 The BDMP Formalism Applied to Attack Modeling

3.1 Foundations

Originally, BDMP are a formalism which combines the readability of classical fault trees with the modeling power of Markov chains [3]. Generally speaking, it changes the fault tree semantics by augmenting it with a special kind of links called triggers, and associating its leaves to Markov processes, dynamically selected in function of the states of some other leaves. This allows for sequences and simple dependencies modeling, while enabling efficient quantifications. The original definition, the mathematical properties and different examples are provided in [3]. In this section, we present the main elements of theory and features offered by a straightforward adaptation of BDMP to security modeling, summing up and completing ref. [16].

The components of BDMP. Informally, "triggered" Markov processes (noted P_i and presented in this section) are associated to the leaves i of an attack tree $\mathcal{A}$. Each process has two modes: *Idle* and *Active* (formally noted 0 and 1). The former models an on-going event, in general an attacker action, the latter is used when nothing is in progress. The mode of a given P_i is a Boolean function of the states of the other processes. Fig. 1 presents the components of a security-oriented BDMP.

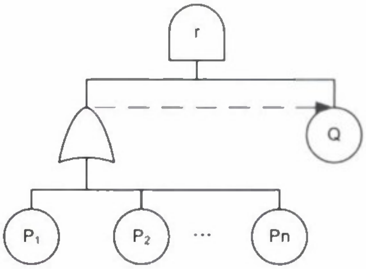


Fig. 1. A small BDMP

More formally, it is a set $\{\mathcal{A}, r, T, P\}$ composed of:

- an attack tree $\mathcal{A} = \{E, L, g\}$, where:
  - $E = G \cup B$, with G a set of logical gates, and B a set of basic security events (e.g. attacker actions), corresponding to the leaves of the BDMP;
  - $L \subset G \times E$ is a set of oriented edges, such that (E, L) is a directed acyclic graph with $\forall i \in G, sons(i) \neq \emptyset$ and $\forall j \in B, sons(j) = \emptyset$, with $E \xrightarrow{sons} P(E), sons(i) = \{j \in E/(i,j) \in L\}$
  - $g : G \rightarrow \mathbb{N}^*$ is a function defining the parameter k of the gates which are all considered to be k/n logical gates ($k = 1$ for OR gates, $k = n$ for AND gates, with n the number of sons)
- r, the final attacker's objective. Formally, it corresponds to a top of (E, L).
- a set of triggers $T \subset (E - \{r\}) \times (E - \{r\})$ such that $\forall(i,j) \in T, i \neq j$ and $\forall(i,j) \in T, \forall(k,l) \in T, i \neq k \Rightarrow j \neq l$. If i is called origin and j target, it means that origin and target of a trigger must differ, and that two triggers cannot have the same target. Triggers are represented by dotted arrows.
- a set P of triggered Markov processes $\{P_i\}_{i \in B}$. Each P_i is defined as a set $\{Z_0^i(t), Z_1^i(t), f_{0\rightarrow 1}^i, f_{1\rightarrow 0}^i\}$ where:
  - $Z_0^i(t)$ and $Z_1^i(t)$ are two homogeneous Markov processes with discrete state spaces. For k in $\{0, 1\}$, the state space of $Z_k^i(t)$ is $A_k^i(t)$. Each $A_k^i(t)$ contains a subset $S_k^i(t)$ which corresponds to success or realization states of the basic security event modeled by the process P_i.
  - $f_{0\rightarrow 1}^i$ and $f_{1\rightarrow 0}^i$ are two "probability transfer functions" defined as follows:
    - for any $x \in A_0^i$, $f_{0\rightarrow 1}^i(x)$ is a probability distribution on A_1^i such that if $x \in S_0^i$, then $\sum_{j \in S_1^i}(f_{0\rightarrow 1}^i(x))(j) = 1$,
    - for any $x \in A_1^i$, $f_{1\rightarrow 0}^i(x)$ is a probability distribution on A_0^i such that if $x \in S_1^i$, then $\sum_{j \in S_0^i}(f_{1\rightarrow 0}^i(x))(j) = 1$.

Triggers and P_is are intimately linked, as the P_is switch instantaneously between modes, via the relevant probability transfer function, according to the state of some externally defined Boolean variables, called process selectors (defined in the next paragraph). The process selectors are defined by means of triggers. Generally speaking, a trigger modifies the mode of the P_i associated to the leaves of the sub-tree it points at, when its origin changes from false to true. The modes are then switched from *Idle* to *Active*, representing the progress of the attacker in the attack scenario possibilities captured by the overall BDMP.

The three families of Boolean functions of time. A BDMP defines a global stochastic process, modeling the evolution of an attack and the dynamic behavior of its perpetrator. Each element i of $\mathcal{A}$ is associated to three Boolean functions of time: a structure function $S_i(t)$, a process selector $X_i(t)$ and a relevance indicator $Y_i(t)$. The three families of these functions are defined as follows (note that to simplify reading, the time t is not indicated but should appear everywhere):

- $(S_i)_{i \in E}$ is the family of structure functions: $\forall i \in G, S_i \equiv (\sum_{j \in sons(i)} S_j \geq g(i))$ and $\forall j \in B, S_j \equiv (Z^j_{X_j} \in F^j_{X_j})$ with X_j indicating the mode in which P_j is at time t. $S_j = 1$ corresponds to the realization of a basic security event (like an attacker action success).
- $(X_i)_{i \in E}$ are the mode selectors, indicating which mode is chosen for each process. If i is a top of $\mathcal{A}$, then $X_i = 1$ else $X_i \equiv \neg[(\forall x \in E, (x,i) \in L \Rightarrow X_x = 0) \vee (\exists x \in E/(x,i) \in T \wedge S_x = 0)]$. This means that $X_i = 1$ except if the origin of a trigger pointing at i has its structure function equal to 0, or if i has at least one parent and all its parents have their process selector equal to 0.
- $(Y_i)_{i \in E}$ are the relevance indicators. They are used to mark the processes to be "trimmed" during the processing of the Markov chain when exploring the possible sequences. Trimming strongly reduces the combinatorial explosion while yielding exact results in our assumptions (cf. the next paragraph and 3.4). If $i = r$ (final objective), then $Y_i = 1$, else $Y_i \equiv (\exists x \in E/(x,i) \in L \wedge Y_x \wedge S_x = 0) \vee (\exists y \in E/(i,y) \in T \wedge S_y = 0)$. This formally says that $Y_i = 1$ if and only if $i = r$, or i has at least one "relevant parent" whose $S_i = 0$, or i is the origin of at least one trigger pointing at an element whose $S_i = 0$.

Mathematical properties. A BDMP can be seen as a robust mathematical formalism thanks to the two following theorems:

Theorem 1. *The functions* $(Y_i), (X_i), (Y_i)$ *are computable for all* $i \in E$ *whatever the BDMP structure.*

Theorem 2. *Any BDMP structure associated to an initial state defined by the modes and the* P_i *states, uniquely defines a homogeneous Markov process.*

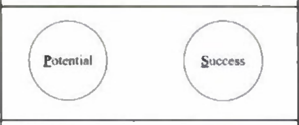
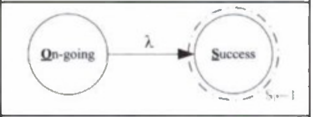
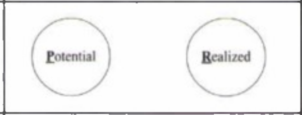
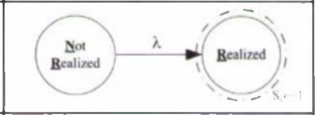
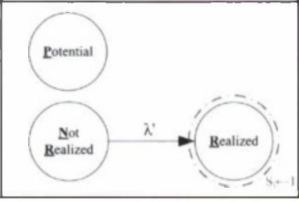
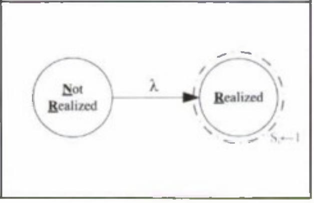
The proof for these theorems can be found in [3]. In addition to their robustness, BDMP allow for a dramatic combinatory reduction by relevant event filtering, thanks to the trimming mechanism associated to the (Y_i) values. This mechanism can be illustrated as follows: in Fig. 1, once a basic security event P_i has been realized, all the other $P_{j \neq i}$ are no longer relevant: nothing is changed for "r" if we inhibit them. The number of sequences leading to the top objective is n if the relevant events are filtered $((P_1, Q), (P_2, Q), ...)$; it is exponential otherwise $((P_1, Q), (P_1, P_2, Q), (P_1, P_3, Q), ...)$.

Theorem 3. *If the (P_i) are such that $\forall i \in B, \forall t, \forall t' \geq t, S_i(t) = 1 \Rightarrow S_i(t') = 1$ (which is always true in our paper), then $Pr(S_T(t) = 1)$ is unchanged whether irrelevant events (with $Y_i = 0$) are trimmed or not.*

The proof of this last theorem is given in [3]. It implies that trimming on the basis of the (Y_i) does not change the quantitative values of interest (cf. 3.4). Moreover, it corresponds to the natural and rational behavior of the attacker.

The basic leaves and their triggered Markov processes. The definition of three kinds of leaves is sufficient to offer large attack modeling capabilities. Their triggered Markov processes are represented informally in Tab. 1.

Table 1. The three basic security leaves for attack modeling

Leaf type & icon	*Idle* Mode (X_i=0)	Transfer between modes	*Active* Mode (X_i=1)
Attacker Action (AA)		P⇔O (with *Pr = 1*) S⇔S (with *Pr = 1*)	
ISE Instantaneous Security Event		P⇒NR (with *Pr=1-γ*) P⇒R (with *Pr = γ*) R⇔R (with *Pr = 1*) P⇐NR (with *Pr = 1*)	
TSE Timed Security Event	 Potential Not Realized λ' Realized $S_i \leftarrow 1$	P⇒NR (with *Pr = 1*) NR⇔NR (with *Pr=1*) R⇔R (with *Pr = 1*)	

- The "Attacker Action" (AA) leaf models an attacker step towards the accomplishment of his objective. The *Idle* mode means that the action has not at this stage been tried by the attacker. The *Active* mode corresponds to actual attempts for which the time needed to succeed is exponentially distributed with a parameter λ. When (X_i) changes from 0 (*Idle*) to 1 (*Active*), the leaf state goes from *Potential* to *On-going*; when (X_i) goes back from 1 to 0, if the attack has not succeeded, the leaf state goes back to *Potential*, if it has succeeded, the leaf comes back to the *Success* state of the *Idle* mode. Formally, the probability transfer functions are: $f_{0\to 1}(P) = \{\Pr(O) = 1, \Pr(S) = 0\}$, $f_{1\to 0}(O) = \{\Pr(P) = 1, \Pr(S) = 0\}$, $f_{1\to 0}(S) = \{\Pr(P) = 0, \Pr(S) = 1\}$.
- The "Timed Security Event" (TSE) leaf models a timed basic security event the realization of which impacts the attacker's progress, but which is not under the attacker's direct control. The time needed for its realization is exponentially distributed. When the leaf comes back to the *Idle* mode, the leaf state

can then be either *Realized* or *Not Realized*, depending on whether the TSE occurred or not in *Active* mode. If unrealized, it is up to the analyst to decide if a realization is then possible in *Idle* mode, by using a $\lambda' \neq 0$. This can be useful when using phased approaches as described in Section 3.3. Formally, the transfer functions are as follows: $f_{0\to1}(P) = \{\Pr(NR) = 1, \Pr(R) = 0\}$, $f_{0\to1}(NR) = \{\Pr(NR) = 1, \Pr(R) = 0\}$, $f_{0\to1}(R) = \{\Pr(NR)=0, \Pr(R)=1\}$, $f_{1\to0}(NR)=\{\Pr(NR)=1, \Pr(R)=0\}$, $f_{1\to0}(R)=\{\Pr(NR)=0, \Pr(R)=1\}$.

- The "Instantaneous Security Event" (ISE) leaf models a basic security event that can happen instantaneously with a probability γ, when the leaf switches from the *Idle* to *Active* mode. In the *Idle* mode, the event cannot occur and the leaf stays in the state *Potential*. In the *Active* mode, the event is either *Realized* or *Not Realized*. State changes are necessarily the result of changes in (X_i). Formally, the probability transfer functions are: $f_{0\to1}(P) = \{\Pr(NR) = 1 - \gamma, \Pr(R) = \gamma\}$, $f_{0\to1}(R) = \{\Pr(NR)=0, \Pr(R)=1\}$, $f_{1\to0}(R) = \{\Pr(NR) = 0, \Pr(R) = 1\}$, $f_{1\to0}(NR) = \{\Pr(P) = 1, \Pr(R) = 0\}$.

3.2 Sequence Modeling

The triggers allow for an efficient and readable modeling of the sequential nature of attacks: often, some actions or events need to be undertaken or realized first before further steps in the attack process can be attempted. Fig. 2 presents a simple example with a sequence of three actions with such a constraint, based on an Operating System (OS) attack. Reference [16] proposes an alternative example, modeling the attack of a Remote Access Server (RAS), while a complete use-case is presented in Section 3.4.

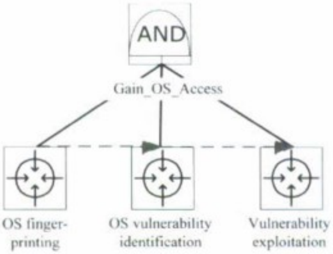


Fig. 2. A simple OS attack

3.3 Concurrent or Exclusive Alternatives

For a given intermediate objective, an attacker may have different alternatives. A natural way of modeling this with BDMP and classical attack trees is with OR gates. Fig. 3 represents two different approaches with an example dealing with OS fingerprinting. On the left side, a simple OR gate is used: passive and active techniques are tried simultaneously, which may not reflect a realistic attacker behavior. Passive techniques, being more discrete, would normally be tried first and, if not successful, given up after some time for active ones. Triggers cannot model such a behavior. "Phase leaves", used on the right side of Fig. 3, allow this behavior to be modeled; their formal definition is given in [16].

3.4 Diverse and Efficient Quantifications: Principles and Use-Case

The interest of BDMP does not only lie in the possibility to represent sequences. They enable diverse time-domain quantifications, including the probability for

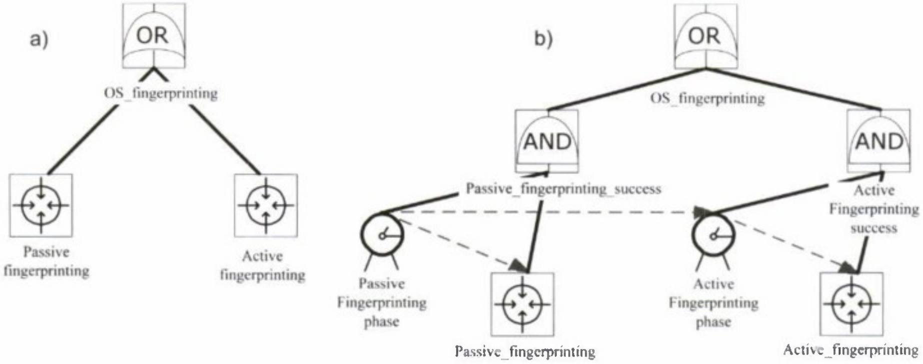


Fig. 3. Modeling parallel or phased alternatives

an attacker to reach his objective in a given time or the overall mean time for the attack to succeed. In addition, BDMP analysis yields the enumeration of all the possible attack paths, ordered by their probability of occurrence in a given time. Such results can be efficiently computed thanks to an original analytical method developed for large Markov models, and thus applicable to BDMP [4]. Indeed, as explained previously, BDMP are high-level representations of potentially large Markov chains; however, the treatment of such chains is usually confronted with state-space explosion. It is overcome using a path-based approach, exploring the sequences leading to the undesirable states. Such an approach enables exact calculations for small models by exhaustive exploration. For larger models, it is possible to obtain controlled approximations by limiting the sequence exploration to those having a probability greater than a given threshold. In both cases, the probability of the explored sequences is computed by the closed form expression given in [5]. Sequence exploration takes advantage of the trimming mechanism described in Section 3.1, which leads to a strong combinatorial reduction.

More concretely, the analyst must define the λ parameters of the exponential distributions and the γ parameters of the ISE leaves. Defining the λs is done by reasoning in terms of Mean Time To Success (MTTS), i.e. $1/\lambda$, like in [9,6,20]. The γs are also set subjectively. The parameters should be estimated based on the intrinsic difficulty of the attacker actions, his estimated skills and resources, and the level of system protection. We have used the KB3 workbench [2] for the model construction and quantitative treatments in this paper. Fig. 4 models the attack of a password-protected file, of which a copy has been stolen. In our scenario, obtaining the password is the only way to access its content, needed by the attacker within a week (this may take place in a call for tender in a competitive environment). The parameters chosen are not given here for space limitation reasons, but they can be found in the technical report [15].

Such parameters lead to a probability of success in a week of 0.422, with an overall MTTS of 22 days. An exhaustive exploration gives 654 possible sequences; Table 2 shows a representative excerpt. The beginning of a phase

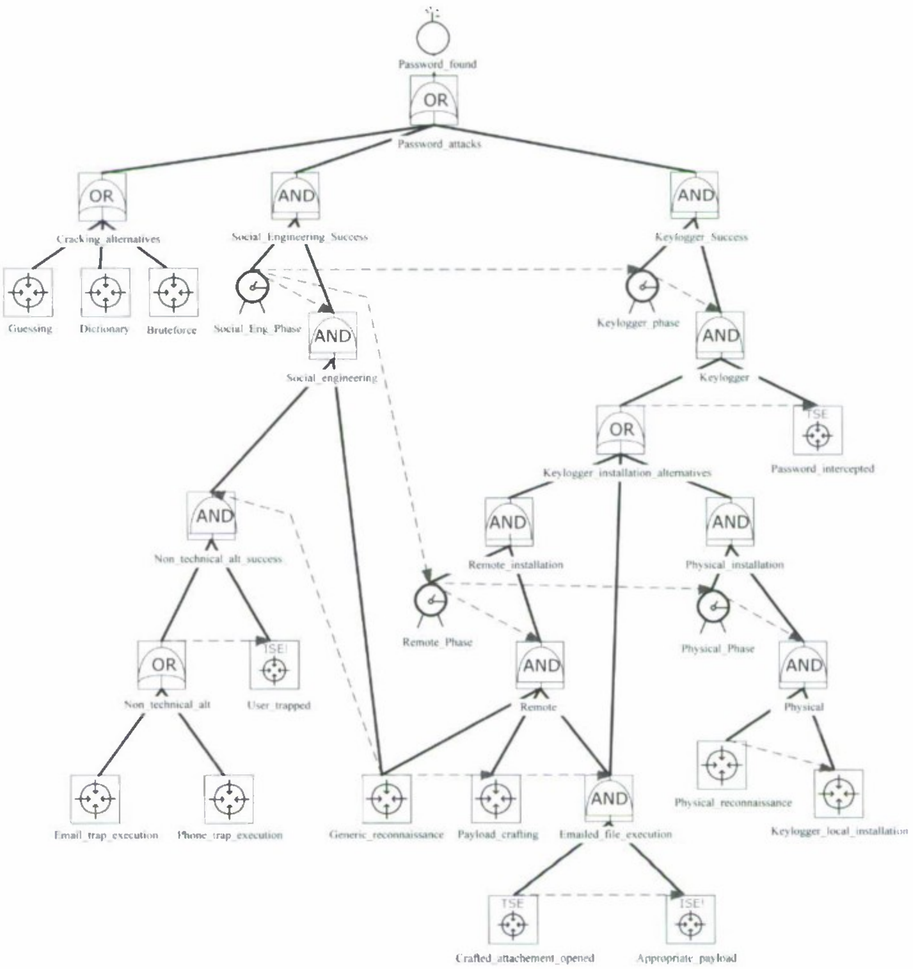


Fig. 4. Attack of a password-protected file

is marked as "<phase>" and its end as "</phase>". Even if phases are not basic security events, they are fully part of the sequences as they structure their chronology. The same applies to the leaves that are realized unnecessarily; they are marked in *italics*. As one can see, most of the sequences include one or more unnecessary actions or events that have no effect on the global success of the attack and as such, these sequences are non-minimal. The minimal sequences are called success sub-sequences, or SSS. Seq. 1 to 4 are minimal and weigh probabilistically 47% of all the sequences. Seq. 5 and 6 are good examples of non-minimal sequences. Bruteforce is a specific leaf as it is also the only single element SSS. It appears directly as a minimal sequence in line 3, but also ends numerous non-minimal sequences. In fact, the consolidated contribution of all

Table 2. Selection of sequences with quantifications

	Sequences	**Probability in a week**	**Average duration**	**Contrib.**
1	<Social Eng>Generic reconn., Email trap exec., User trapped	1.059×10^{-1}	9.889×10^{4}	25.1%
2	<Social Eng>Generic reconn., Phone trap exec., User trapped	5.295×10^{-2}	9.889×10^{4}	12.5%
3	Bruteforce	2.144×10^{-2}	5.638×10^{4}	5.1%
4	<Social Eng></Social Eng><Keylogger><Remote></Remote> <Physical> Physical reconn., Keylogger local installation, Password intercepted	1.749×10^{-2}	2.976×10^{5}	4.1%
5	<Social Eng></Social Eng><Keylogger> <Remote>*Generic reconnaissance* </Remote><Physical>Physical reconnaissance, Keylogger local installation, Password intercepted	1.350×10^{-2}	3.677×10^{5}	3.2%
6	<Social Eng>*Generic reconnaissance, Email trap execution, User trapped(failure)*, Bruteforce	1.259×10^{-2}	2.610×10^{5}	3.0%
...				
20	<Social Eng></Social Eng><Keylogger><Remote>Generic reconnaissance, Payload crafting, Appropriate payload, Password intercepted	2.500×10^{-3}	2.761×10^{5}	0.6%
...				
34	<Social Eng></Social Eng><Keylogger> <Remote>Generic reconn., *Payload crafting* </Remote> <Physical>*Crafted attachement opened, Appropriate payload*, Physical reconn., Keylogger local installation, Password intercepted	1.506×10^{-3}	4.594×10^{5}	0.4%

the sequences ended by bruteforce weighs 40% of all the sequences. Such a strong weight despite bruteforce's large MTTS is due to the absence of other steps to be fulfilled. This points to a more generic statement: a complete analysis should not only use the list of sequences, but also consider complementary views, incl. consolidated contributions of SSS. Seq. 3 to 19 involve only two SSS; seq. 20 relies on a new SSS, then one has to wait until seq. 34 to find another one. This latter sequence illustrates the specificity of TSE leaves, which are able to be realized in Idle mode if the leaf has been Active at least once.

3.5 Hierarchical and Scalable Analysis

It is possible to choose for each attacker action the depth of analysis, leading to different breakdowns depending on the analysis needs. This hierarchical behavior is a powerful property directly inherited from the attack tree formalism. In Fig. 4, the password cracking alternatives have been broken down quite roughly into three techniques which might have been decomposed themselves into much finer possibilities; on the other hand, the social engineering and the keylogger subtrees are slightly more developed. More detailed breakdowns would have been possible. In fact, BDMP with more than 100 leaves are routinely processed in reliability studies [2]: the method is also scalable for security applications.

4 Integrating Defensive Aspects: Detection and Reaction

Holistic approaches to security generally cover protection, detection and reaction. The level of protection can be considered as intrinsically reflected by the BDMP structure, modeling only possible ways for attacks, and its leaves' parameters (λs and γs), reflecting the attack difficulty confronted with a given protection level. This section presents the specifically tailored extensions to BDMP needed to model detection and reaction aspects.

4.1 The IOFA Detection Decomposition

The integration of detection in a dynamic perspective has led us to distinguish four types of detection for the AA and TSE leaves, differentiated by the moment when the detection takes place. Type I (Initial) detections take place at the very start of the attacker actions or of the events modeled; type O (On-going) take place during the attacker attempts or during the events modeled; type F (Final) detections take place at the moment the attacker succeeds in an action or when an event is realized; Type A (A posteriori) detections take place once an action or an event has been realized, based on the traces left by such an action or event.

Each of them has a specific relevance in a security context. Such distinction allows for a fine-tuned and complete modeling of detection; it is designated by the acronym IOFA. ISE leaves have been treated slightly differently with two distinct detections, depending on the realization outcome.

4.2 Extending the Theoretical Framework

In order to model detections & reactions, we extend the framework of § 3.1 by:

- associating to each element a Boolean D_i, called Detection status indicator;
- replacing the *Active* mode by *Active Undetected* and *Active Detected* modes;
- selecting the mode on the basis of X_iD_i, and not only X_i, as described in Tab. 3 (note that in the formal notations of the following sections, 0 in subscript corresponds to the *Idle* mode and covers $X_iD_i = 00$ or 01);
- extending the leaves' triggered Markov processes with new states, transitions, and probability transfer functions, modeling detections and reactions.

Table 3. The new compound process selector X_iD_i and the corresponding modes

X_iD_i	00	01	10	11
Mode	Idle		Active Undetected (AU)	Active Detected (AD)

Detection and reaction in the triggered Markov processes. In this framework, a P_i is a set $\left\{Z_0^i(t), Z_{10}^i(t), Z_{11}^i(t), f^i_{0\to 10}, f^i_{0\to 11}, f^i_{10\to 11}, f^i_{10\to 0}, f^i_{11\to 0}\right\}$ where:

- $Z_0^i(t), Z_{10}^i(t), Z_{11}^i(t)$ are three homogeneous Markov processes with discrete state spaces. For $k \in \{0, 10, 11\}$, the state space of $Z_k^i(t)$ is A_k^i. Each A_k^i contains a subset S_k^i which corresponds to success or realization states of the basic security event modeled by the process P_i, and a subset D_k^i which corresponds to detected states.

- $f^i_{0\to10}, f^i_{0\to11}, f^i_{10\to11}, f^i_{10\to0}, f^i_{11\to0}$ are five "probability transfer functions" defined as follows:
  - for any $x \in A^i_0$, $f^i_{0\to10}(x)$ is a probability distribution on A^i_{10}, such that if $x \in S^i_0$, then $\sum_{j\in S^i_{10}}(f^i_{0\to10}(x))(j) = 1$, and if $x \in D^i_0$, then $\sum_{j\in D^i_{10}}(f^i_{0\to10}(x))(j) = 1$;
  - for any $x \in A^i_0$, $f^i_{0\to11}(x)$ is a probability distribution on A^i_{11}, such that if $x \in S^i_0$, then $\sum_{j\in S^i_{11}}(f^i_{0\to11}(x))(j) = 1$, and if $x \in D^i_0$, then $\sum_{j\in D^i_{11}}(f^i_{0\to11}(x))(j) = 1$;
  - for any $x \in A^i_{10}$, $f^i_{10\to11}(x)$ is a probability distribution on A^i_{11}, such that if $x \in S^i_{10}$, then $\sum_{j\in S^i_{11}}(f^i_{10\to11}(x))(j) = 1$, and if $x \in D^i_{10}$, then $\sum_{j\in D^i_{11}}(f^i_{10\to11}(x))(j) = 1$;
  - for any $x \in A^i_{11}$, $f^i_{11\to0}(x)$ is a probability distribution on A^i_0, such that if $x \in S^i_{11}$ then $\sum_{j\in S^i_0}(f^i_{11\to0}(x))(j) = 1$, and if $x \in D^i_{11}$, then $\sum_{j\in D^i_0}(f^i_{11\to0}(x))(j) = 1$;
  - for any $x \in A^i_{10}$, $f^i_{10\to0}(x)$ is a probability distribution on A^i_0, such that if $x \in S^i_{10}$ then $\sum_{j\in S^i_0}(f^i_{10\to0}(x))(j) = 1$, and if $x \in D^i_{10}$, then $\sum_{j\in D^i_0}(f^i_{10\to0}(x))(j) = 1$.

Note that $f^i_{11\to10}$ is not defined: an attacker once detected cannot subsequently become undetected.

The triggered Markov processes of Section 3.1 are re-engineered to integrate detection and reaction features, as presented in Tab. 4. They support the IOFA detection model of Section 4.1. Transition parameters associated to detection are marked with a "D" in subscript. In the case of the AA and TSE leaves, this letter is followed in parenthesis by the type of detection (I, O, F or A) they characterize; in the case of the ISE leaves, it is followed by the characterized outcome ("/R" in case of realization, "/NR" in case of bad outcome for the attacker). The success and realization parameters are linked to the detection status of the leaf: "/D" in subscript means "having been detected", whereas "/ND" means "having not been detected". Discs with dotted circumferences represent "instantaneous" states whereas full discs are regular timed states. By instantaneous states we mean either:

- Artificial states introduced for the sake of clarity, but which could be removed by merging the incoming timed transitions with the outgoing instantaneous transitions into single timed transitions (e.g. the state SPD in Tab. 4),
- Special "triggering" states which have been introduced to change the D_i values, and trigger mode changes based on internal leaves evolution. For instance in Tab. 4, in AU mode, an arrival either in the "Detected" or the "Success Detected" states triggers an instantaneous mode switch towards the AD mode: both arrivals set the Detection indicator status D_i at 1, passing the Boolean X_iD_i value, used to select the mode, from 10 to 11. Such "triggering" instantaneous states are represented by striped discs.

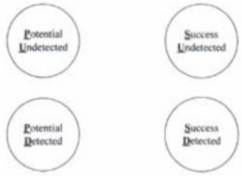
Reaction "propagation". The extended Markov model of the "Attacker Action" leaf in AU mode (cf. Tab. 4) is a good illustration on how detection is taken into account "within" a given leaf, and can provoke a local mode switch towards the AD mode. This changes the leaf parameter $\lambda_{S/ND}$ to a new value $\lambda_{S/D}$, turning the action more difficult or even impossible, if $\lambda_{S/D} = 0$, when the attacker is detected. The same applies for the other leaves. But such mode switches can also be provoked "externally", i.e. by a detection having occurred at the level of a different leaf. In fact, the following possibilities can be distinguished:

- the detection has a strictly local incidence: only the detected attacker action or security event is affected, the rest of the BDMP is unchanged, i.e. the other leaves keep the same parameters λs and γs;
- the detection has an extended incidence, changing not only the on-going detected leaf parameters but also a specific set of other leaves in the BDMP;
- the detection has a global incidence: in case of detection, all the D_i are set to 1, meaning that all the future attacker actions or security events will be in *Detected* mode, with the associated parameters.

This last option is the one that has been adopted in this paper: it is both meaningful in terms of security and straightforward in terms of formalization and implementation. Note that the intermediate option, especially relevant when dealing with multi-domain systems, has been explored by the authors and can be implemented by the introduction of "detection triggers". The associated developments are not given here for space limitation reasons.

Use-case taking into account detections and reactions. The use-case of Section 3.4 has been completed by adding detection and reactions possibilities. The chosen parameters, not given here for space limitation reasons, can be found in [15]. Globally, the introduction of detections and reactions reduces the probability of success within a week by about 14%, from 0.423 to 0.364. This modest reduction can be explained by the fact that the most probable success sequence, the single off-line bruteforce, is not subject to detection. In fact, even with systematic detections and perfect reactions (the attack is stopped), the attacker would still have a 0.201 probability of success, just by the off-line bruteforce attack. In terms of sequences analysis, the number of possible sequences is much higher (4231 vs. 656 in Section 3.4). Tab. 5 gives a selection of sequences with the conventions of Tab. 2; in addition, detections that occurred are indicated in brackets for the relevant leaves. Here again, the top 2 sequences are direct successes of social engineering techniques, followed by the success of a direct bruteforce attack. In the present case, they are followed by several bruteforce terminated non-minimal sequences, before the first sequences based on the trapped email with malicious payload approach appear (seq. 14 and 17). This differs from Tab. 2 in which the sequences based on physical approaches appear first, whereas they are relegated to seq. 20 and further in the present case. This is related to the detection and reaction possibilities associated here to such sequences. In seq. 20, the attacker has failed in his social engineering attempt to

Table 4. The triggered Markov processes of the AA and ISE leaves

<table>
<tr><th colspan="2">Attacker Action (AA)</th></tr>
<tr><th>Markov processes</th><th>Probability transfer functions</th></tr>
<tr><td>Idle ($Z_0^i(t)$)


Potential Undetected
Success Undetected
Potential Detected
Success Detected

</td>
<td rowspan="3">$f^i_{0\to10}(PU)=\{Pr(OU)=1-\gamma_{D(I)}, Pr(D)=\gamma_{D(I)}, Pr(SD)=0, Pr(SU)=0\}$
$(PD)=\{Pr(OU)=0, Pr(D)=1, Pr(SD)=0, Pr(SU)=0\}$
$(SU)=\{Pr(OU)=0, Pr(D)=0, Pr(SD)=0, Pr(SU)=1\}$
$(SD)=\{Pr(OU)=0, Pr(D)=0, Pr(SD)=1, Pr(SU)=0\}$

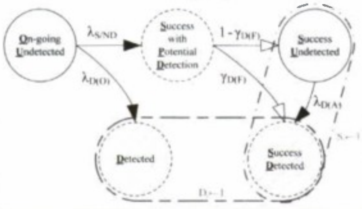
$f^i_{0\to11}(PU)=\{Pr(OD)=1, Pr(SD)=0\}$*
$(PD)=\{Pr(OD)=1, Pr(SD)=0\}$
$(SU)=\{Pr(OD)=0, Pr(SD)=1\}$*
$(SD)=\{Pr(OD)=0, Pr(SD)=1\}$

$f^i_{10\to11}(OU)=\{Pr(OD)=1, Pr(SD)=0\}$*
$(D)=\{Pr(OD)=1, Pr(SD)=0\}$**
$(SD)=\{Pr(OD)=0, Pr(SD)=1\}$**
$(SU)=\{Pr(OD)=0, Pr(SD)=1\}$*

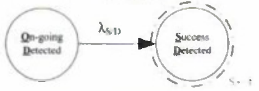
$f^i_{11\to0}(OD)=\{Pr(PU)=0, Pr(PD)=1, Pr(SD)=0, Pr(SU)=0\}$
$(SD)=\{Pr(PU)=0, Pr(PD)=0, Pr(SD)=1, Pr(SU)=0\}$

$f^i_{10\to0}(OU)=\{Pr(PU)=1, Pr(PD)=0, Pr(SD)=0, Pr(SU)=0\}$
$(SU)=\{Pr(PU)=0, Pr(PD)=0, Pr(SD)=0, Pr(SU)=1\}$

* *The detection has occured at a different leaf*

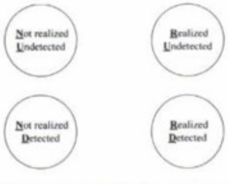
** *Despite D and SD having null durations, these lines are necessary to specify the transfer function, the transfer being potentially triggered by the leaf itself.*</td></tr>
<tr><td>Active Undetected ($Z_{10}^i(t)$)


On-going Undetected
$\lambda_{S/ND}$
Success with Potential Detection
$1-\gamma_{D(F)}$
Success Undetected
$\lambda_{D(O)}$
$\gamma_{D(F)}$
$\lambda_{D(A)}$
S←1
Detected
Success Detected
D←1

</td></tr>
<tr><td>Active Detected ($Z_{11}^i(t)$)


On-going Detected
$\lambda_{S/D}$
Success Detected
S←1

</td></tr>
</table>

<table>
<tr><th colspan="2">Instantaneous Security Event (ISE)</th></tr>
<tr><th>Markov processes</th><th>Probability transfer functions</th></tr>
<tr><td>Idle ($Z_0^i(t)$)


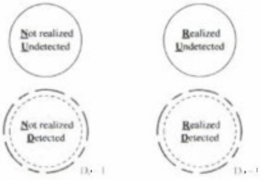
Not realized Undetected
Realized Undetected
Not realized Detected
Realized Detected

</td>
<td rowspan="3">$f^i_{0\to10}(NU)=\{Pr(NU)=(1-\gamma_{S/ND})(1-\gamma_{D/NR}), Pr(RU)=\gamma_{S/ND}(1-\gamma_{D/R}),$
$P(ND)=(1-\gamma_{S/ND})\gamma_{D/NR}, P(RD)=\gamma_{S/ND}\gamma_{D/R}\}$
$(RU)=\{Pr(NU)=0, Pr(RU)=(1-\gamma_{D/R}), Pr(ND)=0, Pr(RD)=\gamma_{D/R}\}$
$(ND)=\{Pr(NU)=0, Pr(RU)=0, Pr(ND)=1-\gamma_{S/D}, Pr(RD)=\gamma_{S/D}\}$
$(RD)=\{Pr(NU)=0, Pr(RU)=0, Pr(ND)=0, Pr(RD)=1\}$

$f^i_{0\to11}(NU)=\{Pr(ND)=(1-\gamma_{S/ND}), Pr(RD)=\gamma_{S/ND}\}$
$(RU)=\{Pr(ND)=0, Pr(RD)=1\}$
$(ND)=\{Pr(ND)=(1-\gamma_{S/D}), Pr(RD)=\gamma_{S/D}\}$
$(RD)=\{Pr(ND)=0, Pr(RD)=1\}$

$f^i_{10\to11}(NU)=\{Pr(ND)=1, Pr(RD)=0\}$
$(RU)=\{Pr(ND)=0, Pr(RD)=1\}$

$f^i_{11\to0}(ND)=\{Pr(NU)=0, Pr(RU)=0, Pr(ND)=1, Pr(RD)=0\}$
$(RD)=\{Pr(NU)=0, Pr(RU)=0, Pr(ND)=0, Pr(RD)=1\}$

$f^i_{10\to0}(NU)=\{Pr(NU)=1, Pr(RU)=0, Pr(ND)=0, Pr(RD)=0\}$
$(RU)=\{Pr(NU)=0, Pr(RU)=1, Pr(ND)=0, Pr(RD)=0\}$</td></tr>
<tr><td>Active Undetected ($Z_{10}^i(t)$)


Not realized Undetected
Realized Undetected
Not realized Detected
D←1
Realized Detected
D←1

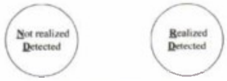
</td></tr>
<tr><td>Active Detected ($Z_{11}^i(t)$)

 </td></tr>
</table>

Table 5. Selection of sequences with quantifications

	Sequences	Probability in a week	Average duration	Contrib.
1	<Social Eng>Generic reconn., Email trap exec., User trapped	1.091×10^{-1}	9.889×10^{4}	30.0%
2	<Social Eng>Generic reconn., Phone trap exec., User trapped	5.456×10^{-2}	9.889×10^{4}	15.0%
3	Bruteforce	2.144×10^{-2}	5.638×10^{4}	5.9%
4	<Social Eng>*Generic reconnaissance*, Bruteforce	1.055×10^{-2}	9.889×10^{4}	2.9%
...	([...], Bruteforce) × 9			
14	<Social Eng><Social Eng><Keylogger><Remote>Generic reconnaissance, Payload crafting(no detection), Appropriate payload(no detection), Password intercepted	2.250×10^{-3}	2.761×10^{5}	0.6%
...	([...], Bruteforce) × 2			
17	<Social Eng>Generic reconnaissance <Social Eng><Keylogger><Remote>Payload crafting(no detection), Appropriate payload(no detection), Password intercepted	1.923×10^{-3}	2.688×10^{5}	0.5%
...	([...], Bruteforce) × 2			
20	<Social Eng>*Generic reconnaissance, Email trap exec., User trapped(failure and detection)* <Social Eng><Keylogger><Remote><Remote> <Physical>Physical reconn., Keylogger local installation, Password intercepted	1.549×10^{-3}	5.991×10^{5}	0.4%

manipulate the user by a forged email and has been detected; the parameters of the subsequent leaves are those corresponding to a detected status. Here again, a complete analysis is not provided, but would benefit from success sub-sequences consolidation views.

5 On-Going and Future Work

A first group of on-going developments aims at supporting security decisions. The new modes related to detection enable new quantifications which may be of interest for the analyst. This includes the mean time to detection (MTTD) or attack sequences classification ordered by their probability of detection. Besides, if the list of sequences provides insightful qualitative and quantitative information, finer-grain analysis, for instance regarding success sub-sequences, are needed to take complete advantage of the model results. Moreover, individual leaf importance factors, adapted to dynamic models as discussed in [13], could be defined for our framework to complete the analyst tool-box. We intend to develop complete and automated tools implementing all these aspects in order to provide a finer and easier support to security decision.

A second type of perspective deals with the BDMP theoretical framework. BDMP have been built on Markovian assumptions and exponential distributions, commonly accepted in reliability engineering [19]. Although such a framework has also been used in security (see [16] for a short review), there is much debate on the appropriate way to model stochastically the behavior of an intelligent attacker, if any. In this perspective, it may be of interest to enable the use of other distributions. This is possible without changing the graphical formalism, but the quantifications could not fully benefit from the methods described in Section 3.4 and would rely on Monte-Carlo simulation.

Finally, the construction of diverse models during this research has led to the identification of recurrent patterns in attack scenarios. A rigorous inventory and categorization of such patterns could lead to a library of small BDMP, modeling classical attack steps ready to assemble when building a complete model.

6 Conclusion

The adaptation and extension of the BDMP formalism offers a new security modeling technique which combines readability, scalability and quantification capability. This paper has presented a complete view of its mathematical framework and has illustrated its use through different use-cases. Sequences, but also concurrent actions or exclusive choices can be easily taken into account. On the defensive side, detection aspects have been integrated while several alternatives are possible for reaction modeling. This extended formalism inherits from the hierarchical and scalable structure of attack trees, allowing different depths of analysis and ease of appropriation, but goes far beyond by taking into account the dynamics of security. It enables diverse and efficient time-domain quantifications, taking advantage of the BDMP trimming mechanism and their associated sequence exploration approach, which have been used extensively in the reliability engineering area. If there is still room for further developments as seen in Section 5, the framework presented here can be already considered as ready to use, bringing an original approach in the security modeling area.

References

1. Amoroso, E.G.: Threat Trees. In: Fundamentals of computer security technology, ch. 2, pp. 15–29. Prentice-Hall Inc., Englewood Cliffs (1994)
2. Bouisson, M.: Automated dependability analysis of complex systems with the KB3 workbench: the experience of EDF R&D. In: Proc. International Conference on Energy and Environment (CIEM 2005), Bucharest, Romania (October 2005)
3. Bouissou, M., Bon, J.: A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes. Reliability Engineering & System Safety 82(2), 149–163 (2003)
4. Bouissou, M., Lefebvre, Y.: A path-based algorithm to evaluate asymptotic unavailability for large Markov models. In: Proc. Reliability and Maintainability Annual Symposium (RAMS 2002), Seattle, USA, pp. 32–39 (2002)
5. Harrison, P.: Laplace transform inversion and passage time distributions in Markov processes. Journal of applied probability 27(1), 74–87 (1990)
6. Jonsson, E., Olovsson, T.: A quantitative model of the security intrusion process based on attacker behavior. IEEE Trans. Soft. Engineering 23(4), 235–245 (1997)
7. Kotenko, I., Stepashkin, M.: Analyzing network security using malefactor action graphs. Int. Journal of Comp. Science and Network Security 6(6), 226–236 (2006)
8. Lippmann, R., Ingols, K.: An annotated review of past papers on attack graphs. Project Report ESC-TR-2005-054, Massachusetts Institute of Technology (MIT), Lincoln Laboratory (March 2005)

9. Littlewood, B., Brocklehurst, S., Fenton, N., Mellor, P., Page, S., Wright, D., Dobson, J., McDermid, J., Gollmann, D.: Towards operational measures of computer security. Journal of Computer Security 2, 211–229 (1993)
10. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006)
11. McDermott, J.P.: Attack net penetration testing. In: Proceedings of the 2000 Workshop on New Security Paradigms, Ballycotton, Ireland, pp. 15–21 (2000)
12. Nicol, D.M., Sanders, W.H., Trivedi, K.S.: Model-based evaluation: From dependability to security. IEEE Trans. Dependable and Secure Comp. 1(1), 48–65 (2004)
13. Ou, Y., Dugan, J.B.: Approximate sensitivity analysis for acyclic Markov reliability models. IEEE Transactions on Reliability 52(2), 220–230 (2003)
14. Patel, S.C., Graham, J.H., Ralston, P.A.: Quantitatively assessing the vulnerability of critical information systems: A new method for evaluating security enhancements. Int. Journal of Information Management 28(6), 483–491 (2008)
15. Piètre-Cambacédès, L., Bouissou, M.: Attack and defense dynamic modeling with BDMP (extended version). Technical Report, Telecom ParisTech, Département INFRES (2010)
16. Piètre-Cambacédès, L., Bouissou, M.: Beyond attack trees: dynamic security modeling with Boolean logic Driven Markov Processes (BDMP). In: Proc. 8th European Dependable Computing Conference (EDCC), Valencia, Spain, pp. 119–208 (April 2010)
17. Piètre-Cambacédès, L., Chaudet, C.: Disentangling the relations between safety and security. In: Proc. of the 9th WSEAS Int. Conf. on Applied Informatics and Communications (AIC 2009), WSEAS, Moscow, Russia (August 2009)
18. Pudar, S., Manimaran, G., Liu, C.: PENET: a practical method and tool for integrated modeling of security attacks and countermeasures. Computers & Security In Press, Corrected Proof (May 2009)
19. Rausand, M., Høyland, A.: System Reliability Theory: Models and Statistical Methods, 2nd edn. Wiley, Chichester (2004)
20. Sallhammar, K.: Stochastic models for combined security and dependability evaluation. Ph.D. thesis, Norwegian University of Science and Technology NTNU (2007)
21. Schneier, B.: Attack trees: Modeling security threats. Dr. Dobb's Journal 12(24), 21–29 (1999)
22. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.: Automated generation and analysis of attack graphs. In: Proc. IEEE Symposium on Security and Privacy (S&P 2002), Oakland, USA, pp. 273–284 (May 2002)

QoS-T: QoS Throttling to Elicit User Cooperation in Computer Systems

Vidyaraman Sankaranarayanan[1], Shambhu Upadhyaya[2], and Kevin Kwiat[3]

[1] Microsoft, 1 Microsoft Way, Redmond, WA 98052
krsna@acm.org
[2] Dept. of CSE, SUNY @ Buffalo, Buffalo, NY 14260
shambhu@cse.buffalo.edu
[3] Air Force Research Laboratory, 525 Brooks Road, Rome, NY 13441
kwiatk@rl.af.mil

Abstract. While there exist strong security concepts and mechanisms, implementation and enforcement of these security measures is a critical concern in the security domain. Normal users, unaware of the implications of their actions, often attempt to bypass or relax the security mechanisms in place, seeking instead increased performance or ease of use. Thus, the human in the loop becomes the weakest link. This shortcoming adds a level of uncertainty unacceptable in highly critical information systems. Merely educating the user to adopt safe security practices is limited in its effectiveness; there is a need to implement a technically sound measure to address the weak human factor across a broad spectrum of systems. In this paper, we present a game theoretic model to elicit user cooperation with the security mechanisms in a system. We argue for a change in the design methodology, where users are persuaded to cooperate with the security mechanisms after suitable feedback. Users are offered incentives in the form of increased Quality of Service (QoS) in terms of application and system level performance increase. User's motives and their actions are modeled in a game theoretic framework using the class of generalized pursuit-evasion differential games.[1,2]

Keywords: Game theory, Human factor in security, Quality of Service, Computer Security, Threat model.

1 Introduction

Traditionally security and quality of service (QoS) have been perceived as only orthogonally achievable goals. The enforcement of security is thought to be a performance obstacle, and guaranteeing QoS is thought to require the relaxation of security mechanisms [4]. These are the misconceptions that drive normal users to bypass or relax the security mechanism in place. Unaware of the implications of their actions, they seek, instead, increased performance or ease of use. Using ineffective passwords

[1] Approved for Public Release; Distribution Unlimited: 88ABW-2008-1165 dated 02 Dec 08.
[2] Work done by first author while at SUNY, Buffalo.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 102–117, 2010.

[2], disabling critical security features, installing untrusted software [5], and not applying security patches in a timely manner are a few instances of user level lapses that impede security. According to a survey [19] conducted by McAfee, users, in the hope of gaining an immediate functionality, may recklessly download and install shareware programs on their company issued laptops or bring in their own gadgets to the workplace. Such lapses are unacceptable in highly critical information systems. However this brings out an interesting point. If the human in the loop proves to be the weakest link, regardless of the sophistication and strength of the security measures taken, their implementation and particularly, their enforcement in a system must be of critical concern. We argue that enforcement of these security measures requires reversing or in the least, manipulating the above misconceptions. Recently there has been some work done on viewing security as one aspect of QoS and in turn, seeking a symbiotic relationship between the two system interests.

In this paper, we aim to exploit the obvious interdependence between quality of service and security in order to improve overall system security, *particularly in interactive systems*. Unlike previous approaches that tend to address a single threat vector [4, 6, 10], the work in this paper describes an underlying approach, similar in theme to [9], that may be used in interactive systems. Given that users prefer greater performance and increased quality of service, we propose a model to prevent security breaches and elicit user cooperation with the security mechanism. The focus of this paper is towards dealing with this class of problems where the system security level is degraded due to user action/inaction. We take the view that all failures due to user action or inaction have to be treated as engineering failures, instead of being ignored. We present a game theoretic model intended to directly counterbalance this risk. The purpose of the model is twofold:

- elicit user cooperation with the security mechanisms in place by gracefully providing incentives to end users as they provide demonstrable evidence of cooperation with the security subsystem
- punish potential intruders who refuse to cooperate with the security subsystem with a reduced QoS

This approach is similar to [7], where hostile users in a wireless ad-hoc network are punished by active jamming. Our approach enjoys two main benefits: It encourages legitimate users to cooperate with security mechanisms as well as deters rogue users by proportionally degrading QoS in light of suspected security breaches.

The underlying concept of degrading performance in case of observed security problems is present in different forms. In the area of network-security, for example, a server may gradually start dropping connections or reducing the QoS to stop a DoS attack or delay the propagation of Worms. We extend this idea to service throttling in order to address the weak human factor. Our mechanism is applied in cases where there is no *absolute certainty* that there is an attack (malicious traffic in the case of a DoS and improper user activity in our case). Degrading performance is done for two reasons: delaying the attack (if there is one in progress) and ensuring user level compliance to the security policies.

The rest of the paper is organized as follows. Section 2 discusses related work on addressing the weak human factor. Section 3 presents the QoS degradation model and the flow of control in the model. Section 4 presents a proof-of-concept simulation that illustrates the usage of this model. Concluding remarks are given in Section 5. The appendices contain the differential games used in the underlying QoS degradation model of Section 3.

2 Related Work

Researchers in [2], [9] and [22] all arrive at the general conclusion that users may be careless and unmotivated when it comes to system security; however they argue that the fault lies ultimately with the design and implementation of these security mechanisms. Adam and Sasse[2] discuss "Users' Perceptions on Security" and the importance of accounting for these perceptions. Dourish et al. [9] argue the importance of creating degrees of security as opposed to the traditional "all-or-nothing" black-box approach. In this way, users naturally distinguish between highly sensitive versus less-sensitive information systems and this manifests itself through different behavior in these different environments. Adam [2] and Sasse[22] also emphasize the importance of *removing* the transparency from security tools, particularly in highly critical systems, and *actively involving users* in the security cycle. With these criteria in mind, in this paper, we developed a graded QoS model to make users personally accountable for the state of the system. Linn [16] introduces a parameter intended to manage the level of protection provided by a security mechanism. Irvine et al. [14, 15] define security as a constructive dimension of QoS rather than an obstacle. Our approach translates variable security levels directly into variable QoS levels returned to the user. In this way, there is a tangible motivation for the user not to circumvent the security mechanism.

The problem of the weak human factor has been researched in the same vein, by using fear appeals [29] or by forcing the user to interrupt their workflow [31] for the 'greater good.' Generic approaches have also been proposed by means of equating safety properties to security properties [3]. Certain online banking systems ask in addition to the password, personal information about the user (like SSN number, Drivers license, etc.) during a login procedure. However such measures are geared only towards malicious users and do not involve legitimate users in the security subsystem. A model called 'safe staging' [30] by Whitten and Tygar extends this notion to legitimate users, where a system restricts the rights of Java applets (the service quality) in response to users' demonstrated understanding of the security implications. As users become more familiar with the security issues, the service quality is increased. Our model extends this notion a step further by incorporating a monitoring and feedback control mechanism to involve legitimate users in a constructive manner.

3 QoS Throttling (QoS-T) Model

Essentially, the problem we seek to solve is an important one, but has eluded a technical solution due to a variety of reasons. Primary among them is the *act of interference* and *lack of control*; any technological solution that seeks to remedy the weak human factor does so by means of either interfering in the workflow of the user or taking away control of the system from the user, or a combination of both factors. These two factors irk users; security designers have not found the correct balance or an alternative. Our approach is a combination of these two factors, but in a very *gradual and subtle manner* with appropriate feedback, thereby giving the user complete control at every stage, with minimum to zero interference to the workflow. The nature of this problem involves understanding and quantifying user actions, their incentives and ensuring an optimal state where the user objectives are met and the system security is also maintained. Thus, the problem may be viewed as one of balancing the objectives of the user and that of the system. In such a situation, game theoretic models apply naturally.

3.1 Why Game Theory?

We have chosen the class of generalized pursuit-evasion differential games for modeling this problem. Game theory helps model a set of 'selfish' and 'rational' players who act in a setting solely for their own advantage. Users in our setting can be said to act selfishly to improve their own QoS. Game theoretic models have been used to infer the incentives of attackers [20] based on their perceived incentives. In this work, we use a game theoretic model to provide incentives to the user in order to elicit cooperation. The purpose of the game theoretic model is to derive a measurable quantity out of the user's actions that can be given as a feedback to the security mechanism. User's actions in the game theoretic setting are equivalent to strategies of a player. The security mechanism can use the payoff function of the game to adjust the QoS. The advantage of modeling the user/resource/security-mechanism scenario as a differential game is that it allows for a flexible definition of the act of "a user accesses a resource." While this definition is abstract at the level at which this model is described, it can be properly interpreted and applied on the specific security domain where its application is relevant. The model is self enforcing; we do not make any assumptions about the coordination between the players of the game. Users in a system need not be aware of the model, nor are they required to consciously participate in any 'game.' Differential game-theory also has the notion of 'continuous' play, which makes it conducive to use it in situations as these. Lastly, the usage of game theoretic notions allows us to specify notions of strategy or best responses of the participating players (the users and the system) thereby leading to good mechanism design that elicits cooperation from users as a natural process. The reader is referred to [8, 17] for a more detailed exposition on game theory. The specifics of the games used in our model are described in the appendix.

3.2 Types of Users

The threat posed by legitimate users in an organization has appropriately been labeled as "The Enemy Within" [19] in a recent survey by McAfee Corporation (http://www.mcafee.com). We can divide the user broadly into two different categories:

- Type I: A Legitimate User – This category of users includes legitimate and authorized users of the system. These users log into the system and execute workflow processes according to their roles. According to the McAfee Survey [19], such users are varyingly labeled as "The Security Softie", "The Gadget Geek" or "The Squatter." While they do not have any stated intentions to disrupt the system, their actions nonetheless endanger the system. For example, these users do not have any idea of the threat model of the system and hence, may not implement the best practices suggested by the organization.

- Type II: A Legitimate, but Malicious User – Similar to Type I users, users in this category are legitimate, i.e., they possess authorized credentials to log into the system. However, their goal is to disrupt the system, either through a self inflicted cataclysmic system compromise or through slow poisoning attacks like leaking confidential information about the organization to its competitors. According to the Survey [19], such users are labeled as "The Saboteur."

Let us first examine the challenges that researchers and designers face when dealing with the weak human factor. In any system, users perform actions towards fulfilling their roles. The notion of actions is an abstract one that can be generalized to most, if not all, systems. Actions can be split in the following manner.

- *Action Type I* – the fundamental user actions required for the workflow: These fundamental actions are defined by the user's role in the environment. For example, a graphics designer will need to use some photo/video editing software. In addition, a device like a tablet may need to be connected to the computer via the USB interface for rendering hand sketches.
- *Action Type II* – Ancillary actions required for the fundamental actions to work: For example, exploring the hard drive is a prerequisite for most job roles. In addition, connecting USB devices, burning images onto a CD may be in this list for a graphics designer.
- *Action Type III* – These are actions that are not predefined like Action types I and II. These actions are the ones that users normally execute without any restrictions, since they do not fall under the purview of 'restricted objects.' They might have the potential to disrupt the working of the system, or may be inimical to the individual. Examples of such actions include clicking on a potential phishing link in an un-trusted/unsigned email.

For those actions that are relevant to the security of the system, there exists an easy or an efficient manner of performing them. For example, choosing a password is an (one time) action that users have to perform when registering into the system. The easy way is to choose a password that is easy to remember (and hence easy to guess/crack). The efficient way, on the other hand, is to choose a complex password that is tough to remember. Similar is the situation with security updates; it is easy to ignore them while it is efficient to update the system. For reasons that are mostly context and domain specific, users prefer to perform only the easy action, and not the efficient one. Viewing the interaction between the user and the system as a set of easy vs. efficient actions, where the easy action is most often the inefficient one, provides us a global view to look into this issue. Thus the main challenge for human centered security schemes is to ensure that users perform the efficient action with awareness of the consequences of their actions. Viewing these actions under the three prisms provides us one methodology to address the human factor related security issues.

3.3 Process Flow

The flow of eontrol for the QoS throttling model is shown in Figure 1. We first start with the seeurity meehanism and derive its requirements. Towards this, we may use the systems' best practices as a guide. Coneurrently, we define the user's workflow proeess. These two steps are one-time proeesses whieh ensure that (a) at no cost is the users workflow adversely affeeted and (b) the Monitoring agent is aware of the security subsystem's expeetations. The user's session then proceeds as usual, where every user action is first filtered by the seeurity polieies in the system. These filtered aetions are monitored by the monitoring agent, whieh deeides if the actions are in eonformance with the security subsystems requirements. If the user is cooperative, he is rewarded with a gradual inerease in the QoS. If the user is not eooperative, a feedback is given (similar to 'Install Updates' dialog box in the windows environment, etc.) with a request to cooperate. If the user still blatantly refuses to cooperate, the gradual application and context speeifie QoS throttling is initiated.

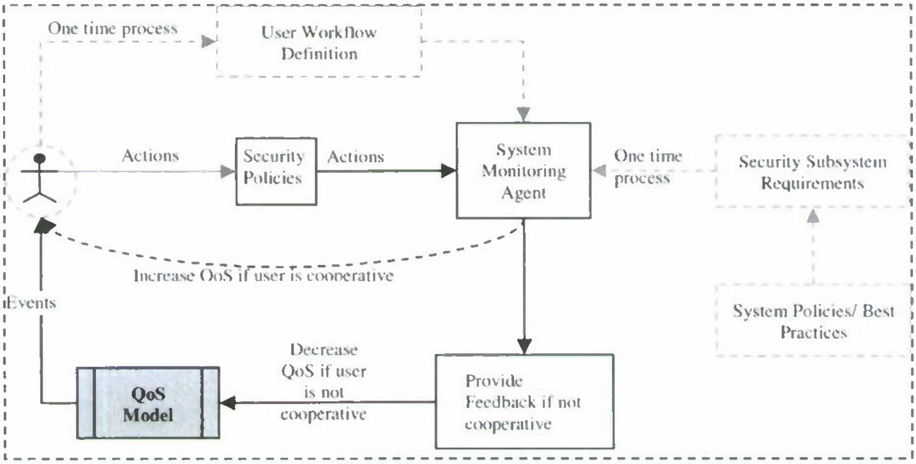

One time process
User Workflow Definition
Actions
Security Policies
Actions
System Monitoring Agent
One time process
Security Subsystem Requirements
Increase QoS if user is cooperative
Events
System Policies/ Best Practices
Decrease QoS if user is not cooperative
QoS Model
Provide Feedback if not cooperative

Fig. 1. Control Flow

3.4 QoS-T Model 1: Exponential Back-Off

Given a singleton process, we discuss a simplistie exponential back-off model to evaluate a decreasing time delay. This time delay could be used as a parameter to the artificial sleep statements (or any other eontext specific delay). This model is useful in situations where the system (and its threat model) is simple enough with an automated meehanism that elassifies the user. We define the QoS throttle through a simple equation:

$$f(x) = (1 - x)\cdot e^{1/x} \qquad : x \in [0,1] \tag{1}$$

where x is the quantitative input that grades the users elassification and $f(x)$ the time delay (in some appropriate time units) that is imposed by the system. The value x can be the trust level of the user in the system, a real number between 0 and 1, where 0 represents a untrustworthy user and 1 a trustworthy user. For those systems that have a mechanism to detect their security level (or trust level of users), the exponential back-off model may be used. For example, the *Compensatory Trust Model*[28] is an automated trust evaluation mechanism specifically designed for users in an authenticated system.The exponential back-off model has the advantage of simplicity, clear intuition and an easy translation to an implementation. Also, the intuition behind it may be changed depending on the system (and the users) to derive other ancillary models (different distributions) that may perform better for particular systems.

3.5 QoS-T Model 2: Game Theoretic Approach

Given a workflow process with multiple sub-processes, we present a game theoretic model that can be used to gradually reduce the QoS of a sub-process and tag the user as proceeding gradually from a non-cooperative user to a malicious user during the workflow.

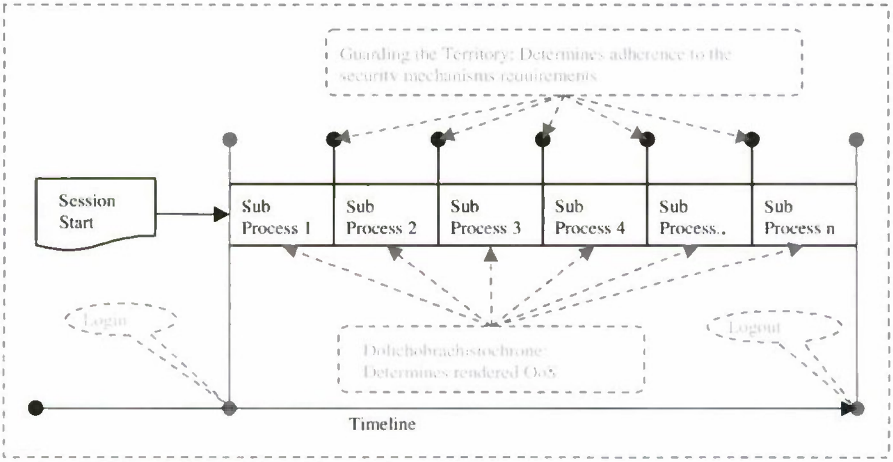

Guarding the Territory: Determines adherence to the security mechanisms requirements
Session Start
Sub Process 1
Sub Process 2
Sub Process 3
Sub Process 4
Sub Process..
Sub Process n
Login
Dolichobrachistochrone: Determines rendered QoS
Logout
Timeline

Fig. 2. Modeling the Workflow and constituent Sub-Processes

We leverage on two well-studied problems of game theory – the two player differential game of "Guarding a Territory" [12] and the "Dolichobrachistochrone" [11]. The entire timeline of the users' actions in a session is split into many fragments. Each fragment represents some sub-process in the workflow (e.g., the execution of a process/application). The process of changing the QoS is equivalent to varying the rate/difficulty with which the user can access the resource or complete execution of the process. The Dolichobrachistochrone game models each fragments of the timeline, i.e., the resource-access event. After every resource access, we need to determine the security state of the system (or in other words, classify the user and hence infer the security state of the system). The second differential game of "Guarding the territory" models the security state of the system. This game also models the point where the *noncompliant user becomes a malicious user.* These two games are chained to provide proper feedback so that the output of the Dolichobrachistochrone game is the input to "Guarding the territory." This process is shown in Figure 2. The

Dolichobrachistochrone game is also called a supergame, within which repetitions of the smaller 'guarding the territory' game is played. The game-theoretic model is more involved, with a greater emphasis on system specifics and an inbuilt mechanism for user classification. A detailed description of these games is given in the appendix.

The control variables of the two games are chosen depending on the system under consideration and the security mechanism. The final step in the model is to chain the two games so that the output of the Dolichobrachistochrone is the input of the "Guarding the territory" game. The payoff of the Dolichobrachistochrone is *x(T)*, which is the distance traveled by the particle P. The player **u** in the "Guarding the territory" game can now travel a distance *x(T)* towards the region **Ω**by a predetermined angle.If the playerwere to reach the region before the session is over, the security state of the system can be changed and appropriate action can be initiated.

4 Proof-of-Concept Illustration

In this section, we introduce a simplified, yet generalized scenario encountered by network administrators in most IT organizations. We then derive the threat model from a basic social engineering attack and present the application of the game theoretic QoS-T model to this problem, illustrating its practical utility.

4.1 Threat Scenario

An experiment [26] was conducted by "The Training Camp" where commuters in London were offered free CD's with special Valentine Day's promotion. Despite a clear warning, most employees apparently inserted the CD and ran the program (which displayed a warning against such actions). In a similar vein, another experiment [25] (akin to a social engineering penetration testing) revealed that free USB disks which were 'discovered' by employees were blindly inserted into computers, thereby triggering the execution of a (potentially malicious) program.

The situation is similar with the case of downloading and installing programs from the Internet. For example, consider the process of downloading and executing a file from the Internet. The user launches a browser, connects to the web site that hosts the file (or is redirected to the site), downloads the executable and then executes it. Assume that the systems' best practices state that unless a downloaded executable is signed by a trusted publisher, it is preferable to not execute it. This typical sequence of operations initiated by the user can be broken down into sub-processes, each of which plays a role in the complete operation. This example brings out the following points:

(a) The process of downloading and running untrusted executables is a manifestation of the weak human factor.
(b) This process is not part of the user's workflow in the organization.
(c) The entire process can be split into a number of sub-processes:
    a. Browsing to an untrusted zone
    b. Initiating a File download
    c. Executing the file

For the sake of illustration, we assume that executing untrusted executables in the current user context is not completely prohibited, but is undesirable.With this scenario, let us explore how the new paradigm can be applied. We have two levels here. First we want to throttle the service quality, but not affect the user's legitimate workflow. Secondly, we would like to use the additional CPU cycles gained to perform some useful work, in terms of increasing system performance and security. To degrade the application level QoS, the browser could be slowed down in a number of ways (inserting artificial sleep statements in the browser process, slowing down the network bandwidth available to the browser process, etc.). This degradation is initiated only after a proper feedback is provided to the user, warning him to refrain from the actions. This degradation by no means affects the system performance (if there are any background processes running) and provides the developer an opportunity to insert (for example) security logging statements, like logging the site where the browser navigated to, the plug-ins activated by the site, etc. After the application has been downloaded, the user could be given an option to run the application inside a sandbox with restricted permissions, or run the application with less than normal privileges. Additionally, the application level QoS could be degraded by inserting artificial sleep statements in an approach similar to [30].

4.2 Threat Model: Multiple Untrusted Applications Execution

In this threat model, we envisage a scenario where users are required to specify in their workflow patterns the most commonly used applications in a typical session. This represents a secure and controlled environment such as the military operations or a secure and compartmentalized job in an industry. As mentioned in Figure 1, specifying the workflow is a onetime process. If the applications users execute fall within the purview of the workflow, they are accorded a high application level QoS. As they execute applications outside the workflow specification (possibly due to malicious intent or due to an impersonation attack), the QoS is gradually reduced. When the number of applications outside the workflow specification exceeds a limit defined by the users trust level, the user is declared malicious. This clearly illustrates the game theoretic model; one game is used to determine the QoS and the other is used to determine when the non-cooperative user becomes a malicious user.

The QoS degradation for this threat modelis similar to the concept of penalizing specific system processes, which is the approach by Somayaji and Forrest [24], where an exponentially increasing delay (artificial sleep statements) was introduced between system calls.

The final step is translating the model to the actual timing details. We fixed the distance of the user from the territory **Ω** (Figure 6 in the appendix) to be the maximum number of unauthorized applications for a standard user (x_o = 7 units in Eq. 5 in the appendix). As we shall see, users are tagged malicious if their trust level is low or never tagged as malicious if their trust level is high, even if they exceed the maximum value set for the standard user. The users' trustworthiness (U_T) was varied between 0 and 1 ($0 < U_T \leq 1$). ω in Eq. 5 in the appendix was set to 1. Finally the delay time (T) in the Dolichobrachistochrone game is inversely proportional to U_T ($T = \alpha / U_T$). One time unit is set to 10 milliseconds for this plug-in. These assignments finally reduced the model to evaluation of the Dolichobrachistochrone game Eq. 5 in the appendix, which now reads as follows:

$$x(T) = x_o - \frac{T}{2} + T - \frac{1}{2}\sin(T) \qquad (2)$$

The variable α (in $T = \alpha / U_T$) for each action was set and subsequently increased according to Table 1. Subsequent values of x_o were assigned to the previous values of *x(T)* as calculated by Eq. 5 in the appendix. We varied the users' trust level and plotted the time delay as well as the number of unauthorized applications it would take for the user to execute to penetrate the territory. The resulting action by the security subsystem depends on the domain. The plug-in raised an administrative alert when the territory was reached by the user.

Table 1. Values of U_T and corresponding α

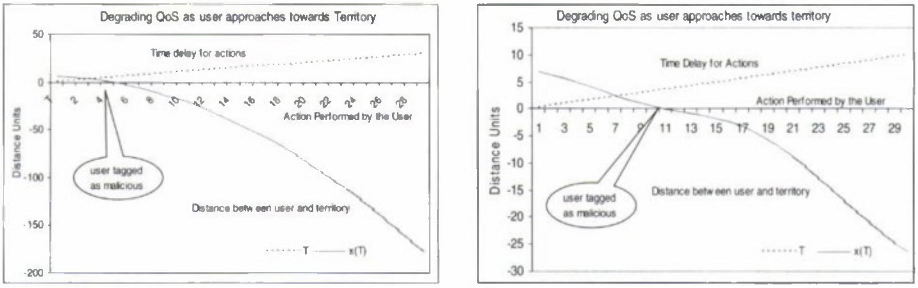
Trust Level (U_T)	**Alpha** (α in $T = \alpha / U_T$)	**Figure**
0.1	Initially set to 0.1 and increased by 0.1	3.a
0.3	Initially set to 0.1 and increased by 0.1	3.b
0.7	Initially set to 0.15 and increased by 0.15	4.a
1.0	Initially set to 0.2 and increased by 0.2	4.b

Figure 3 shows the time delay for low values of user trust levels and the number of actions (or equivalently, the number of untrusted applications executed) it takes for the users to transit from a non-cooperative user to a malicious user.

QoS Degradation for U_T = 0.1: Figure 3.a shows the time delay rate (T) and the progress of the user towards the territory (*x(T)*) for a trust level of 0.1. Since the users trust level is very low, the user rapidly progresses towards the territory, indicative of his low trust level; he is tagged as malicious (at the point where *x(T)* crosses y = 0) by the fifth unauthorized application.

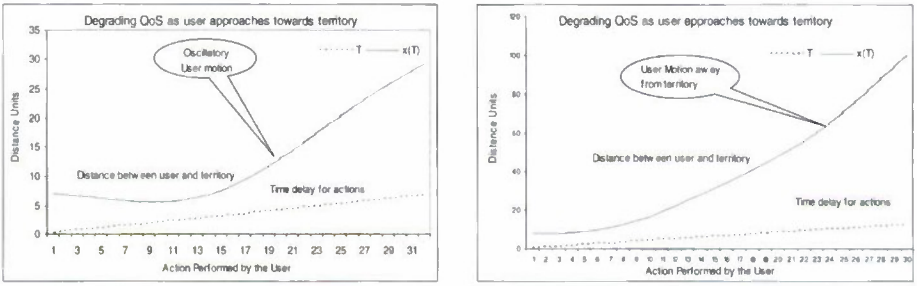
QoS Degradation for U_T = 0.3: Contrast this with Figure 3.b, which shows the same plot for a trust level of 0.3. The time delay rate is still the same (α is the same), but the user approaches the territory slowly, indicative of an increased trust level, and is tagged as malicious only by the 11th unauthorized application, as opposed to the fifth one in Figure 3.a. Figure 4 shows the time delay for high values of user trust levels. In this case, we note that the user does not actually penetrate the territory, indicative of the high trust level. Instead, there is a gradual oscillatory movement due to the sinusoidal component in Eq. 5 in the appendix.

QoS Degradation for U_T = 0.7: As illustrated in Figure 4.a, the user initially approaches the territory since the session scope is not complied with. Due to the sinusoidal component in Eq. 5 in the appendix, the initial approach towards the territory is replaced with a movement away from the territory. We interpret this sinusoidal oscillation as follows: This user is not deemed malicious at any point of time, due to his high trust level. But, we also set *a lower time delay rate* as the system expects him to be cooperative and security conscious due to his high trust level. For example, the time delay for this user at the end of the 11th unauthorized application action is 2.35 time units (23.5 milliseconds) while the time delay for the user in Figure 4.b (U_T = 0.3) is 3.66 time units (36.6 milliseconds).



(a) QoS degradation for $U_T = 0.1$ (b) QoS degradation for $U_T = 0.3$

Fig. 3. QoS degradation for low values of user trust level



(a) QoS degradation for $U_T = 0.7$ (b) QoS degradation for $U_T = 1$

Fig. 4. QoS degradation for high values of user trust level

QoS Degradation for $U_T = 1$: Here we illustrate a situation where the user is completely trustworthy. In Figure 6.b, the user does not even approach the territory (indeed, he moves away from it) since he is completely trustworthy. However, the *time delay is made higher* (0.2 units). The time delay at the end of the 11th action for this user is 4.4 units (44 milliseconds). Such progressively high time delays virtually render the unauthorized applications inoperable (for the user). Note that the act of setting higher time delays for highly trusted users is intuitive and logical, since trusted users in mission critical areas are expected to be aware of the security subsystem and hence, cooperative. Their high trust levels ensure that they are not tagged as malicious at any point in time, *a privilege they earn at the cost of actively cooperating with the security subsystem.* A simple scheme to ensure that users consistently approach the territory as time progresses is to provide a feedback loop and lower their trust level progressively. This aspect, however, is out of scope of this paper, for trust assignment and management is another research area by itself.

5 Conclusion and Future Work

Farmer's Law states, "The security of a computer system degrades in direct proportion to the amount of use the system receives." Farmer self-proclaimed this as his law in a survey on the security of Key Internet Hosts, highlighting the fact that users are often the greatest risk to system security. In a similar vein, Schneier[23] states that the very interaction between humans and systems forms the greatest risk to IT systems. For specific threat models (like weak passwords, phishing, social engineering, etc.), there are specific solutions. But the greater problem of involving the users in the security loop has remained unaddressed so far. The reason for this lies in the misconception that QoS and security are orthogonally achievable goals. In highly-critical information systems the need to appeal to these users and elicit their cooperation is paramount. Trading application level QoS in terms of transparency/ease of usage for user involvement with the security mechanisms in place is justified and in fact, necessary. The solution advocated in this paper is a graceful degradation of the rendered application specific QoS that the user perceives *in the face of a conspicuous lack of cooperation*. For example, consider the case of data breaches in corporate environments; a recent article in the Wall Street Journal states [32] states that data breaches are on the rise; most often, the data breaches are not detected immediately and offenders are rarely, if ever, held accountable. The QoS-T framework proposed in this paper could be viewed as a contractual requirement by the customer of businesses; it may be viewed as a mechanism to correct complacency by corporate members' in-situ. Any complacency by businesses (and their employees) in applying appropriate security measures towards data protection would lead to a lowering of QoS, which in turn, would directly affect productivity (and hence, would affect the "sacred" bottom-line). Thus conformance to security measures will not be limited to merely a moral code but enforced with a monetary means.The application of game theoretic models to practical scenarios will lead us into interesting problems [18] which have to be resolved in a context specific manner. Although it is debatable if such a model will *really* ensure user cooperation, we hope that in the same manner a user types his password carefully the second time to avoid typographical mistakes (and hence additional delays in password systems), the implementation of this model will encourage the user to cooperate and actively participate with the security subsystem.

References

1. DoD Directive 8500.1, Information Assurance, IA (2002)
2. Adams, A., Sasse, M.A.: Users are not the enemy. Commun. ACM 42, 40–46 (1999)
3. Brostoff, S., Sasse, M.A.: Safe and Sound: a Safety-Critical Approach to Security. In: Proceedings of the workshop on New security paradigms. ACM Press, Cloudcroft (2001)
4. Brostoff, S., Sasse, M.A.: Ten strikes and you're out: Increasing the number of login attempts can improve password usability. In: Workshop on Human-Computer Interaction and Security Systems, Ft. Lauderdale, FL, USA (2003)
5. CERT, CERT® Advisory CA-2000-04 Love Letter Worm (2005), `http://www.cert.org/advisories/CA-2000-04.html`
6. Hinds, C., Ekwueme, C.: Increasing security and usability of computer systems with graphical passwords. In: Proceedings of the 45th annual southeast regional conference. ACM Press, Winston-Salem (2007)
7. Levin, D.: Punishment in Selfish Wireless Networks: A Game Theoretic Analysis. In: Proceedings of Economics of Networked Systems. NetECON Ann Arbour, Michigan (2006)
8. Davis, M.: Game Theory: A nontechnical introduction. Dover, New York (1983)

9. Dourish, P., Grinter, R., Dalal, B., Flor, J.D., Joseph, M.: Security Day-to-Day: User Strategies for Managing Security as an Everyday, Practical Problem, Institute for Software Research, University of California, Irvine (2003)
10. Bergadano, F., Gunetti, D., Picardi, C.: User authentication through keystroke dynamics. ACM Trans. Inf. Syst. Secur. 5, 367–397 (2002)
11. Freedman, A.: The Dolichobrachistochrone Game, Differential Games, 107p. John Wiley & Sons, Inc., Chichester (1971)
12. Freedman, A.: Guarding a Territory, Differential Games, 29p. John Wiley & Sons, Inc., Chichester (1971)
13. Howard, M.: Browsing the Web and Reading E-mail Safely as an Administrator. In: MSDN (2004)
14. Irvine, C., Levin, T., Sypropoulou, E., Allen, B.: Security as a Dimension of Quality of Service in Active Service Environments. In: International Workshop on Active Middleware Services, San Francisco, CA (2001)
15. Irvine, C., Levin, T.: Quality of Security Service. In: Proceedings of the New Security Paradigms Workshop. ACM Press, Ballycotton (2000)
16. Linn, J.: Generic Security Service Application Program Interface, IETF Request for Comments (1993)
17. Luce, R.D., Raiffa, H.: Games and Decisions. Dover, New York (1989)
18. Mahajan, R., Rodrig, M., Wetherall, D., Zahorjan, J.: Experiences applying game theory to system design. In: Proceedings of the ACM SIGCOMM workshop on Practice and Theory of Incentives in Networked Systems. ACM Press, Portland (2004)
19. McAfeeCorporation, The Enemy Within (2005), http://www.theregister.co.uk/2005/12/15/mcafee_internal_security_survey/
20. Liu, P., Zang, W., Yu, M.: Incentive-based modeling and inference of attacker intent, objectives, and strategies. ACM Trans. Inf. Syst. Secur. 8, 78–118 (2005)
21. Sankaranarayanan, V., Chandresekaran, M., Upadhyaya, S.: Position: The User is the Enemy. In: Proceedings of the New Security Paradigms Workshop, New Hampshire, USA (2007)
22. Sasse, M.A.: Computer Security: Anatomy of a Usability Disaster, and a Plan for Recovery. In: CHI 2003 Workshop on Human-Computer Interaction and Security Systems, Ft. Lauderdale, FL, USA (2003)
23. Schneier, B.: Secrets and Lies: Digital Security in a Networked World. John Wiley & Sons, Inc., New York (2000)
24. Somayaji, A., Forrest, S.: Automated Response Using System-Call Delays. In: Usenix Security Symposium (2000)
25. Stasiukonis, S.: Social Engineering, the USB Way. Dark Reading, Secure Network Technologies Inc. (2006), http://www.darkreading.com/document.asp?doc_id=95556&WT.svl=column1_1
26. Sturgeon, W.: Proof: Employees don't care about security, Silicon.com (2006), http://software.silicon.com/security/0,39024655,39156503,00.htm
27. Tzur, R.: SandboxIE (2006), http://www.sandboxie.com/
28. Sankaranarayanan, V., Upadhyaya, S.: A Trust Assignment Model based on Alternate Actions Payoff. In: Stølen, K., Winsborough, W.H., Martinelli, F., Massacci, F. (eds.) iTrust 2006. LNCS, vol. 3986, pp. 339–353. Springer, Heidelberg (2006)
29. Weirich, D., Sasse, M.A.: Pretty good persuasion: a first step towards effective password security in the real world. In: Proceedings of New Security Paradigms Workshop. ACM Press, Cloudcroft (2001)

30. Whitten, A., Tygar, J.D.: Safe staging for computer security. In: HCI and Security Systems Workshop, CHI, Ft. Lauderdale, Florida (2003)
31. Xia, H., Brustoloni, J.C.: Hardening Web browsers against man-in-the-middle and eavesdropping attacks. In: Proceedings of the 14th international conference on World Wide Web. ACM Press, Chiba (2005)
32. Wall Street Journal: Data Breaches Surpass 2007 Level, But Businesses Rarely Are Penalized (2008)

Appendix

Dolichobrachistochrone: The Dolichobrachistochrone game is a two player differential game where a point mass P in a uniform gravitational field is constrained to move without friction along a given curve γ. This is illustrated in Figure 5. For equation convenience, the gravitational field is in the direction of the positive y axis. The objective of P is to choose a curve so that it reaches the line x = 0 (the y-axis) in minimum time. The player E has an objective of trying to slow P as much as possible. E has a force ψ that can be applied to slow P from reaching x = 0. The conditions of the game dictate that the particle P will definitely reach the y axis in a finite time. P's objective is to minimize its arrival time to x = 0. E's objective is to maximize the time for P to reach x = 0. In our model, P represents the user and E the security mechanism. For every access to a resource, the user attempts to minimize his time of access. This translates to P minimizing its arrival time to x = 0. The security mechanism (E in the game) attempts to vary the rendered QoS according to the force ψ. Figure 5.a shows the particle P falling through the curve γtowards the y axis (x = 0). Figure 5.b shows the player E with an opposing force ψ. The equations of motion for the particle P are described in [11]. The payoff of the game is the distance traveled by the particle P.

$$P(\psi) = x(T) \tag{3}$$

where T is the time for P to reach x = 0 and ψ is a positive constant (ψ = EB = EC in Figure 5.b).

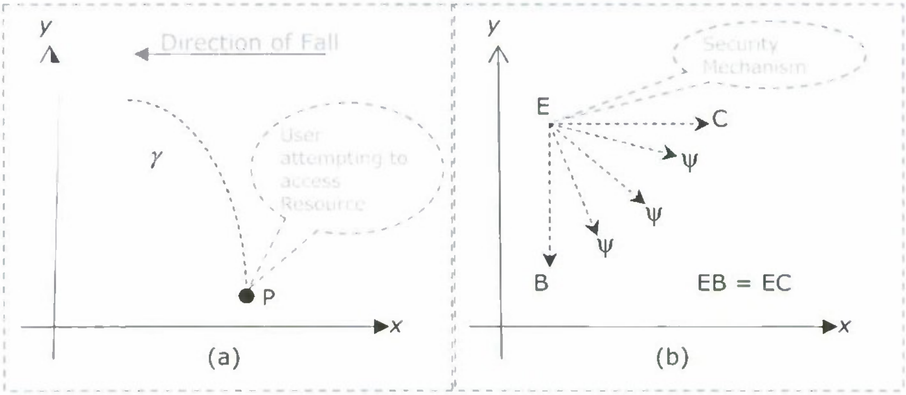


Fig. 5. Dolichobrachistochrone Game

The optimal trajectory for the particle P is given by:

$$y(t) = \frac{y(T)}{2}(1 + \cos\frac{(T-t)}{\sqrt{y(T)}}) \qquad (4)$$

$$x(t) = x_o - \frac{\sqrt{y(T)}}{2}t + \omega t + \frac{y(T)}{2}\sin\frac{T-t}{\sqrt{y(T)}} - \frac{y(T)}{2}\sin\frac{T}{\sqrt{y(T)}}\,;\ y_o = y(0) \qquad (5)$$

wherc -1 ≤ ω ≤ 1. The readcr is referred to [11] for more details. The Value of the game (which is the payoff under optimal conditions) is *x(T)* that is evaluated from Eq. 5. Hence the security mechanism can choose *T* (which is the time (delay) taken by the user to access the resouree) or equivalently, the foree ψ based on the system parameters like the value of the resource being pursued ($\mathbf{R}_v$), the trustworthiness of the user ($\mathbf{U}_T$), etc.

Guarding the Tcrritory: This gamc represents a model in whieh a player v is guarding a territory **Ω**against an invasion by the player **u**, asshown in Figure 6. The motion of **u** and **v** are deseribed by differential equations [12]. Thc initial eonditions are set as **x**(0) = A and **y**(0) = B. As illustrated in Figure 4, player **v**, the Seeurity mechanism, is loeated at B, while player **u**, the user, is located at A. In Figure 6, the players are initially separated by a distance AB. C is the mid-point of segment AB. CY* is perpendicular to AB, with Y* being the nearest point to the region **Ω**sueh that Y*Z* is perpendieular to Y*C. Z* is the point on the region **Ω** that is nearest to the line segment CY*. We denote the distance of any point **x** on the plane to the territory **Ω** as *d*(**x**,**Ω**). Each eooperative action by the user symbolically takes him farther away from the region **Ω**. The modcl expects the security mechanism to provide a feedback on the nature of the user's action and a quantitative measure of the same (whieh is obtained, in this case, from the Doliehobrachistochrone game).

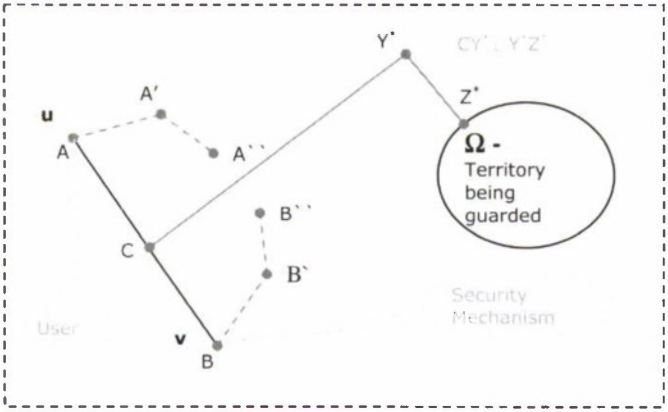


Fig. 6. Guarding the lerrilory

This measure effectively takes the user towards the region **Ω** or away from it. The region **Ω** is the "intrusive" region which the security mechanism can be thought of as trying to protect. The payoff function of the differential game is given as:

$$P(\mathbf{u},\mathbf{v}) = \begin{cases} d(\mathrm{x}(\tau), \Omega), & \text{if } \tau<\mathrm{T} \\ \mathrm{N}, & \text{if } \tau = \mathbf{T} \text{ and } \mathrm{x}(\tau) \text{ lies on the same side of } \mathrm{CY}^{*} \text{ as A} \\ 0, & \text{if } \tau = \mathbf{T} \text{ and } \mathrm{x}(\tau) \text{ lies on the same side of } \mathrm{CY}^{*} \text{ as B} \end{cases} \quad (6)$$

where N >d(Y*, **Ω**). In a typical equilibrium strategy, every motion of **u** towards the region**Ω** is matched by **v** by a similar mirror image move across CY* as indicated by the dotted lines in Figure 4. For instance, the move AA\` is matched by BB\`, A\`A\`\` by B\`B\`\`. The objective of the player **u** is to minimize the payoff P(**u**,**v**) in Eq. 5, whereas player **v** tries to maximize it. Hence, in the original game, if player **u**chose not to come near the territory **Ω**, he is penalized by a payoff N. If **v** did not guard the territory "very well", he is penalized by a payoff of 0.

Problems of Modeling in the Analysis of Covert Channels

Alexander Grusho[1], Nikolai Grusho[2], and Elena Timonina[2]

[1] Moscow State University, GSP-2, Leninskie Gory,
Moscow, 119992, Russian Federation
grusho@yandex.ru

[2] Russian State University for the Humanities, 25 Kirovogradskaya,
Moscow, 113534, Russian Federation

Abstract. Sometimes the analysis of covert channel is weakly dependent on the correctness of probabilistic models, but more often the result of such analysis is seriously dependent on the choice of a probabilistic model. We show how the problem of detection of covert communications depends on the correctness of the choice of probabilistic model. We found[1] the dependence of judgments about invisibility of covert communication from the bans in a probabilistic model of the legal communication.

Keywords: covert channel, detection of information flow, covert communication.

1 Introduction

Applications of statistical methods in the analysis of covert channels or covert communication are widely discussed in the literature [1,2]. Then everywhere we'll use the term covert channel. In the analysis of covert channels the focus is concentrated on statistical detection of communications with hidden information [3,4,5]. Another important task is to assess the capacity of covert channels [6,7]. All works, which are associated with these tasks, use probabilistic models of legal communications and probabilistic models of information hiding.

In this paper we discuss some problems of probabilistic modeling in the analysis of covert channels. Sometimes the analysis of covert channels is weakly dependent on the correctness of probabilistic models. We show that the problem of detection of covert channels depends strongly on the correctness of the choice of probabilistic model. We note that the small differences in that probabilistic models can significantly affect the topological structure of the sets, associated with supports of probability measures, used in the simulation. We show that the topological structure of such sets depends on the bans (prohibitions) of certain configurations which cannot appear in legal communications.

[1] This work was supported by the Russian Foundation for Basic Research, grant 10-01-00480.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 118–124, 2010.

Research of such bans received little attention in the analysis of covert channels. The appearance of forbidden configurations can greatly simplify the search of covert channels.

The paper has the following structure. In section 2 we consider the motivation examples to explain the problem of modeling of covert channels. Section 3 shows how to build a topological space, whose structure reflects the properties of probabilistic models of covert channels. We find some topological properties of some sets related to the restrictions in the structures of legal communications. We have built an example in which the appearance of a ban severely alters the topological structure of the considered sets. In conclusion we summarize the results and outline the path for further research.

2 Preliminary Discussion

Let T - a channel from computer A to computer B. All transmitted information in T must not contradict business processes in A and B, which T supports. A' is an agent of an adversary in A and B' is an agent of an adversary in B. A' may suspend the transmission of legal information and inserts its own message. The schedule of hidden message transfer is known to B'. It is a key.

We consider the Simmons's model [8] for hidden signal (message) from A' to B', having the channel (network) T at their disposal. Observer U intercepts everything that is transmitted from A to B and decides if there is a covert communication or not. When A chooses messages independently then the case was considered in [9]. There we proved conditions of absolute invisibility of covert channel.

Let X be the set of possible messages. Regardless of what methods of analysis U has, his decision is based on a set $S \subseteq X$ of messages that U considers as covert communications. If x is the message, which U has observed, then U makes the decision on the existence of hidden transmission in the case, when $x \in S$. That is, U has a computable for him function $\pi(x, S)$, where $\pi(x, S)$ - an indicator of the set S.

If x does not belong to the set S, then U does not consider the transfer of x as illegal, i.e., U "does not see" the threat of x.

Suppose that A' and B' know S. Then the hidden transfer must be constructed so that the transmitted message $x \notin S$. We say that in this case U "does not see" a covert channel. This approach describes both as deterministic so and probabilistic models of covert channels. For deterministic models it is obvious.

Consider a probabilistic model of a covert channel. It presupposes the existence of a probability measure P_0 on X, which corresponds to a legal choice of message x. That is, U tests the hypothesis $H_0 : P_0$ by the observation x. Criterion is defined by the critical set S, rejecting the hypothesis H_0 with a given level of error probability ϵ (i.e. $P_0(S) \leq \epsilon$).

A with the "help" of A' chooses a message x in accordance with his probability measure P_1. For example A' uses pauses between the sentences to introduce its own information. In this case B receives the message x, and B', knowing the secret key k, uses the distribution $P_1(x|k)$ to get information like in [10]. For example B' counts the posteriori probability to make a decision about covert information transfer out of secretly determined periods of time. Here $P_1(S)$ - the probability to be noticed when sending covert messages.

There are two approaches in constructing models of action of the participants A', B' and U. The first approach is typical for U. He builds any sort of model. In this model, U builds critical set S, and decides if there is or there is no a covert channel. If U guessed and correctly solved the problem of protection, then it does not matter how correct is his model. He succeeded because his model has suggested him the way to success. Such an approach to model in the analysis of covert channels we'll call "lodestar".

The second approach in constructing models of action of the participants is typical for A'. When building a covert channel, he must explain the invisibility of the transmission, because defeat could follow severe consequences for him. To justify the invisibility of the covert communication A' and B' must have an adequate model of the set S.

Consider the problem of the adequacy of the model of a covert channel. If U knows that for any choice of the key distributions P_0 and P_1 coincide, it is obvious that he has no criterion for identifying the covert channel [2]. We explain this in the following example.

Example 1. Participants A' and B' suggest that the model of legal transfer through the channel T is determined by the distribution P_0. Transfer in the channel T is a realization of an infinite sequence of iid random variables taking values in the alphabet $\chi = (x_1, ...x_m)$ with probabilities $p = (p_1, ...p_m)$, $p_i > 0$, $i = 1, ..., m$, $\sum_{i=1}^{m} p_i = 1$. A' may do insert into legal transmission. In this model A' can easily prove that the following transmission scheme is absolutely invisible for U.

Let the set of messages of participant A' consists of $L_1, ..., L_N$. He encodes them as follows. A' builds N random sequences $\Delta_1, ..., \Delta_N$ in accordance with the measure P_0, and each message $L_i, i = 1, ..., N$, is associated with $\Delta_i, i = 1, ..., N$. In addition, he builds a long key random binary sequence G. To arrange invisible channel for U, this set of data secretly from the U is transmitted to B'. Let A' is going to transmit message L_j. On the place, where the sequence G has 1, A' additionally inserts the next character of the sequence Δ_j. We obtain thus a sequence of iid random variables with the $(p_1, ...p_m)$ -distribution of characters on the set χ. Thus, for U we have $P_0 = P_1$ and U does not see the hidden transfer.

Example 2. Assume, for simplicity that in the previous example, $m = 2, \chi = \{0, 1\}$ and the legal channel is generated by the simple homogeneous Markov chain, in which the transition probability matrix has all elements positive except $P(1, 1) = 0$.

The covert communication using the method of the previous example can be easily detected. It may be done even in the case when sequences $\Delta_j, j = 1, ..., N$, are chosen in accordance with the same Markov chain as in the legal communication. In fact, in the sequence $\Delta_j, j = 1, ..., N$, there is an infinite number of 1, and the probability, that before places of incorporating the hidden elements it will always be element equals to 0, tends to 0. Hence, with probability tending to 1, there will be the combination (1, 1) which is outlawed in the Markov chain. When such a combination is obtained, U identifies the covert transmission.

These examples show that a mistake in the choice of measure P_0 may lead to detection of covert channel constructed by A'.

More difficulties for A' in the construction of a covert channel occur, if the distributions of sequences, transmitted through the channel (network) T, may vary. That is, instead of P_0 the distribution of the legal transmitted sequence belongs to the family $\{P_\lambda, \lambda \in \Lambda\}$, and the choice of a critical set S for U may depend on λ. It means that U receives from A additional information about the distribution in the channel (network) T. In this case A' does not possess such information.

3 Asymptotical Case

Consider the problem of constructing invisible covert channel in an asymptotic formulation. Let $X_i, i = 1, 2, ...$, be the sequence of finite sets, the time is discrete. At each n U observes the vector $\boldsymbol{x}_n = (x_1, ..., x_n)$, $x_i \in X_i$. We can assume that the maximal information available to U is an infinite sequence $\boldsymbol{x}$ in the space of all possible messages X:

$$\boldsymbol{x} \in \prod_{i=1}^{\infty} X_i = X.$$

In accordance with the previous assumptions, there are several models of legal communications defined by the probability measures P_λ, $\lambda \in \Lambda$, on X as functions on σ-algebra $\mathcal{A}$, which is generated by cylindrical subsets of X. Denote $P_{\lambda,\, n}$, $\lambda \in \Lambda$, $n \in N$, be the projections of measures P_λ on the first n coordinates of the sequences of X. Denote $D_{\lambda,\, n}$ be the support of the measure $P_{\lambda,\, n}$ in the space $\prod_{i=1}^{n} X_i$, and

$$\Delta_{\lambda,\, n} = D_{\lambda,\, n} \times \prod_{i=n+1}^{\infty} X_i.$$

In the asymptotic formulation of the problem of covert communication detection U has a sequence of criteria $t_{\lambda,\, n}$, $\lambda \in \Lambda$, $n \in N$, that are specified by a sequence of critical sets

$$S_{\lambda,\, n} \subseteq \prod_{i=1}^{n} X_i.$$

U chooses n and uses $S_{\lambda,\, n}$ to make his decision about the presence of a covert channel in $\prod_{i=1}^{n} X_i$ with a known $P_{\lambda,\, n}$. Namely, if the sequence $\boldsymbol{x}_n$ belongs to $S_{\lambda,\, n}$, then U announces the identification of a covert channel. We believe that $S_{\lambda,\, n}$ chosen so that

$$P_\lambda(S_{\lambda,\, n}) \to_{n \to \infty} 0.$$

for any $\lambda \in \Lambda$. This means that U asymptotically identifies the correct legal transfer. Obviously, the covert channel is not seen for U for any λ, if the choice of hidden messages does not belong to $\bigcup_{\lambda \in \Lambda} S_{\lambda,\, n}$ for any n.

If it is not known how U selects a critical set, then A' must comply with certain rules for the selection of hidden messages.

1. At a certain $\lambda \in \Lambda$ for every n hidden message should belong to $D_{\lambda,\, n}$. For U it is reasonable to include in $S_{\lambda,\, n}$ all the sequences $\boldsymbol{x}_n$ from $\prod_{i=1}^{n} X_i$ for which

$$P_{\lambda,\, n}(\boldsymbol{x}_n) = 0.$$

This rule is illustrated in Example 2.

2. If

$$F = X \setminus \left[\bigcup_{n} \bigcup_{\lambda \in \Lambda} \left(S_{\lambda,\, n} \times \prod_{i=n+1}^{\infty} X_i \right) \right].$$

becomes empty, that will make impossible for A' to choose a hidden message. If F consists of a finite number of sequence, then U increases a total number of forbidden messages by adding last several messages from F. In the case, when F consists of a countable set of sequences, then U can decrease F to a sufficiently large finite set, and significantly limit the ability to hide information.

Let's investigate the effect of these rules on the choice of A'. The discrete topology can be considered on X_i. Then X becomes a topological space (Tychonoff product [11,12]). This space is compact, because X_i is finite. In addition, the topological space X has a countable base, because class of cylindrical sets is countable. In our case, the Borel σ-algebra $\mathcal{B}$ coincides with $\mathcal{A}$. Therefore, all measures P_λ, $\lambda \in \Lambda$, are defined on $\mathcal{B}$.

It is obvious that for any $\lambda \in \Lambda$ the sequence $\Delta_{\lambda,\, n}$, $n \in N$, is nonincreasing, since

$$D_{\lambda,\, n-1} \times X_n \supseteq D_{\lambda,\, n}.$$

Hence, there is a limit

$$\Delta_\lambda = \bigcap_{n=1}^{\infty} \Delta_{\lambda,\, n}.$$

All $\Delta_{\lambda,\, n}, n \in N$, are closed and open sets of a topological space X. Therefore, for all $\lambda \in \Lambda$ sets Δ_λ are closed in the Tychonoff product. Then Δ is a closed set:

$$\Delta = \bigcap_{\lambda \in \Lambda} \Delta_\lambda = \bigcap_{\lambda \in \Lambda} \bigcap_{n=1}^{\infty} \Delta_{\lambda,\, n}. \tag{1}$$

In formula (1) we can take a countable set of nonincreasing cylindrical sets $\Delta_n, n \in N$, such that

$$\Delta = \bigcap_{n=1}^{\infty} \Delta_n.$$

There are several cases.

Case I. $\Delta = \emptyset$. Because by the compactness of X there exists M, for which

$$\bigcap_{n=1}^{M} \Delta_n = \emptyset.$$

By nonincreasing of sequence $\Delta_n, n \in N$,

$$\Delta_M = \emptyset.$$

This means that for some n A' has no choice for invisible covert communication. Any choice of A' can be seen by U in these circumstances. I.e. for A' there is no guarantee of invisibility of the hidden transfer.

Case II. The set Δ includes some open set of X. Since any open set in X is a countable union of some of cylindrical sets, then all sets Δ_λ, $\lambda \in \Lambda$, include at least one cylindrical set which is common for all sets Δ_λ. Any cylindrical set is uncountable. Consequently, in Δ_λ there exists an uncountable set of points, each of them has measure 0. Therefore, A' can choose of them a satisfactory sequence (a finite set of sequences) for covert communication, ensuring the nonexistence of a consistent sequence of criteria for detection this communication by U. And there is a consistent procedure for B' to understand the information that A' sent to him.

Case III. If the conditions I or II are not fulfilled, then the topological structure of a closed set Δ is more complex and requires further study.

Example 3. Consider the family of distributions (P_0, P_2), corresponding to the examples 1 and 2. That is, P_0 corresponds to a sequence of iid random variables taking values 0 and 1 with probabilities $1-p$ and p, $0 < p < 1$. P_2 corresponds to a stationary homogeneous Markov chain on the states 0 and 1 with a matrix P, where $p_{11} = q_1, p_{12} = 1 - q_1, p_{21} = q_2, p_{22} = 1 - q_2$.

If $0 < q_1 < 1$ and $0 < q_2 < 1$, then for any n, the equality $D_{0,\,n} = D_{0,\,n}$ is fulfilled. Therefore $\Delta = \Delta_0 = \Delta_2 = X$. In this case Δ contains an open set. As was shown [13] A' can choose any sequence as a hidden signal. In this case for U there is no consistent sequence of criteria for identifying a covert channel.

Consider the case when $0 < q_1 < 1$, as $q_2 = 1$. In this case, for any n $D_{0,\,n} \supset D_{2,\,n}$. So $\Delta = \Delta_2$ and Δ is an uncountable closed set without interior points. Thus we have conditions of the case III.

4 Conclusion

We found the dependence of judgments about invisibility of covert channels and the choice of a probabilistic model of the legal communication. When we try to simplify the probabilistic models of legal communications we may loose the bans, which are prohibited in legal communications. Bans may generate significant changes in the topological structure of certain subsets of supports of probability measures. These structural changes can be detected by a computer simulation that may give a chance to simplify the search of bans in real systems.

References

1. Johnson, N.F., Duric, Z., Jajodia, S.: Information Hiding: Steganography and Watermarking-Attacks and Countermeasures. Kluwer Academic Publishers, Boston (2000)
2. Wang, Y., Moulin, P.: Perfectly secure steganography: Capacity, error exponents, and code constructions. IEEE Transactions on Information Theory, Special Issue on Security 54(6) (2008)
3. Grusho, A., Kniazev, A., Timonina, E.: Detection of Illegal Information Flow. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 235–244. Springer, Heidelberg (2005)
4. Filler, T., Fridrich, J.: Complete characterization of perfectly secure stego-systems with mutually independent embedding operation. In: Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 19–24 (2009)
5. Cachin, C.: An information-theoretic model for steganography. Information and Computation 192(1), 41–56 (2004)
6. Moulin, P., Wang, Y.: New results on steganographic capacity. In: Proceedings of the Conference on Information Sciences and Systems, CISS, March 17-19 (2004)
7. Filler, T., Fridrich, J., Ker, A.D.: The square root law of steganographic capacity for Markov covers. In: Delp, E.J., Wong, P.W., Memon, N., Dittmann, J. (eds.) Proceedings SPIE, Electronic Imaging, Security and Forensics of Multimedia, San Jose, CA, vol. XI, pp. 18–21 (January 2009)
8. Simmons, G.J.: The prisoners problem and the subliminal channel. In: Chaum, D. (ed.) Advances in Cryptology: Proceedings of Crypto 1983, pp. 51–67 (1984)
9. Grusho, A.: On existence of subliminal channels. Discrete Mathematics and Applications 9(2), 1–8 (1999)
10. Shannon, K.: The works on information theory and cybernetics. Foreign Literature, Moscow (1963) (in Russian)
11. Bourbaki, N.: Topologie Generale. Science, Moscow (1968) (in Russian)
12. Prokhorov, U.V., Rozanov, U.A.: Theory of probabilities. Science, Moscow (1993) (in Russian)
13. Grusho, A., Grebnev, N., Timonina, E.: Covert channel invisibility theorem. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) Proceedings of Fourth International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2007, pp. 187–196. Springer, Heidelberg (2007)

Policy-Based Design and Verification for Mission Assurance*

Shiu-Kai Chin[1], Sarah Muccio[2], Susan Older[1], and Thomas N.J. Vestal[2]

[1] EECS Department, Syracuse University, Syracuse, New York 13244, USA
[2] Air Force Research Laboratory, Rome, New York 13441, USA

Abstract. Intelligent systems often operate in a blend of cyberspace and physical space. Cyberspace operations—planning, actions, and effects in realms where signals affect intelligent systems—often occur in milliseconds without human intervention. Decisions and actions in cyberspace can affect physical space, particularly in SCADA—supervisory control and data acquisition—systems. For critical military missions, intelligent and autonomous systems must adhere to commander intent and operate in ways that assure the integrity of mission operations. This paper shows how policy, expressed using an access-control logic, serves as a bridge between commanders and implementers. We describe an access-control logic based on a multi-agent propositional modal logic, show how policies are described, how access decisions are justified, and give examples of how concepts of operations are analyzed. Our experience is policy-based design and verification is within the reach of practicing engineers. A logical approach enables engineers to think precisely about the security and integrity of their systems and the missions they support.

Keywords: policy, concept of operations, access control, logic.

1 Introduction

Cyber space and physical space are ever more intertwined. Cyber-physical systems, i.e., systems with tight coordination between computational and physical resources, operate in these intertwined worlds. Automatic pilots in aircraft and smart weapons are examples of cyber-physical systems where the capability to complete Boyd's *observe-orient-decide-act* decision loop [1] in milliseconds without human intervention is essential.

For commanders, fulfilling the missions entrusted to them is of paramount importance. As autonomous cyber and cyber-physical systems have by their very nature little, if any, human supervision in their decision loops, mission assurance and mission integrity concerns require that the trustworthiness of these systems be rigorously established.

A practical concern is how commanders and implementers will communicate with each other. Commanders operate at the level of policy: what is permitted and under what circumstances. Implementers are concerned with mechanisms. Our observation is that commanders and implementers communicate through descriptions of policy and concepts of operation. Our key contribution is a methodology for describing policies and trust assumptions within the context of concepts of operations.

* Distribution Statement A—Approved for Public Release—Distribution Unlimited Document #88ABW-2010-0819, dated 24 February 2010.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 125–138, 2010.

The remainder of this paper is organized as follows. First, we informally describe the central elements of policy and concepts of operation that we wish to describe and justify rigorously. Second, we describe the syntax and semantics of our access-control logic. Third, we describe a hypothetical concept of operations, formalize its description, and provide a formal justification for its operations. Finally, we offer summary remarks and conclusions.

2 Elements of Policy and Concepts of Operation

Policies are principles, guides, contracts, agreements, or statements about decisions, actions, authority, delegation, credentials, or representation. Concepts of operation (CONOPS) describe a system from the user's perspective. CONOPS describe the goals, objectives, policies, responsibilities, jurisdictions of various authorities, and operational processes.

The elements of policy we are concerned with include:

- who or what has control over an action and under what circumstances,
- what are recognized tokens of authority,
- who are recognized delegates,
- what credentials are recognized,
- what authorities are recognized and on what are they trusted, and
- any trust assumptions used in making decisions or judgments.

We conceptualize CONOPS as a chain of statements or requests for action. These requests are granted or rejected based on the elements of policy listed above. This is illustrated in Figure 1. What Figure 1 shows is an abstract depiction of a CONOPS that has three or more principals or agents: *P1*, *P2*, and *P3*. Principals are entities such as subjects, objects, keys, tokens, processes, etc. Principals are anything or anybody that makes requests, is acted upon, or is used as a token representing a principal.

CONOPS begin with a statement or request *s1* by *P1*. In the syntax of the access-control logic we introduce next, this is the formula $P1$ says $s1$. Principal *P2*, is envisioned to receive the statement $P1$ says $s1$, and within the context of jurisdiction statements, policy statements, and trust assumptions, *P2* concludes $s2$ is justified. As a result of this justification, principal *P2* transmits a statement $P2$ says $s2$ to principal *P3*, who then reacts within the context of its jurisdiction and policy statements, and trust assumptions. We repeat this for all principals and processes in the CONOPS.

Within the boxes labeled *Principal 2* and *Principal 3* are expressions

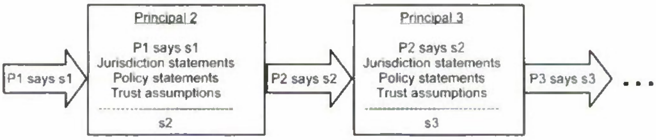


Fig. 1. Concept of Operations

$$\frac{\begin{array}{c} P1 \textsf{ says } s1 \\ \textit{Jurisdiction statements} \\ \textit{Policy statements} \\ \textit{Trust assumptions} \end{array}}{s2} \quad \text{and} \quad \frac{\begin{array}{c} P2 \textsf{ says } s2 \\ \textit{Jurisdiction statements} \\ \textit{Policy statements} \\ \textit{Trust assumptions} \end{array}}{s3}.$$

What the above expressions intend to convey is that based on: (1) the statements or requests $s1$ and $s2$ made by principals $P1$ and $P2$, and (2) the statements of jurisdiction, policy, and trust assumptions under which principals $P2$ and $P3$ operate, $P2$ and $P3$ are logically justified (using the logic and calculus we describe next) to conclude $s2$ and $s3$. As we will see after formally describing the syntax and semantics of our logic, the two expressions above have the form of *derived* inference rules or *theorems* in our calculus. Each step of a CONOPS expressed in this fashion is a theorem justifying the behavior of a system.

One of the principal values of using the access-control logic is the evaluation of a CONOPS for logical consistency within the context of given policies, certifications, and trust assumptions. The process we outline here makes explicit underlying assumptions and potential vulnerabilities. This leads to a deeper understanding of the underpinnings of security and integrity for a system. This greater understanding and precision, when compared to informal descriptions, produces more informed design decisions and trade-offs.

In the following section, we define the syntax and semantics of the access-control logic and calculus.

3 An Access-Control Logic and Calculus

3.1 Syntax

Principal Expressions. Let P and Q range over a collection of principal expressions. Let A range over a countable set of simple principal names. The abstract syntax of principal expressions is:

$$P ::= A \;/\; P\&Q \;/\; P \mid Q$$

The principal $P\&Q$ ("P in conjunction with Q") is an abstract principal making exactly those statements made by both P and Q; $P \mid Q$ ("P quoting Q") is an abstract principal corresponding to principal P quoting principal Q.

Access Control Statements. The abstract syntax of statements (ranged over by φ) is defined as follows, where P and Q range over principal expressions and p ranges over a countable set of *propositional variables*:

$$\begin{array}{c} \varphi ::= p \;/\; \neg\varphi \;/\; \varphi_1 \wedge \varphi_2 \;/\; \varphi_1 \vee \varphi_2 \;/\; \varphi_1 \supset \varphi_2 \;/\; \varphi_1 \equiv \varphi_2 \;/ \\ P \Rightarrow Q \;/\; P \textsf{ says } \varphi \;/\; P \textsf{ controls } \varphi \;/\; P \;\textsf{ reps } Q \textsf{ on } \varphi \end{array}$$

Informally, a formula $P \Rightarrow Q$ (pronounced "P speaks for Q") indicates that *every* statement made by P can also be viewed as a statement from Q. A formula P controls φ is syntactic sugar for the implication (P says φ) $\supset \varphi$: in effect, P is a trusted authority with respect to the statement φ. P reps Q on φ denotes that P is Q's delegate on φ; it is syntactic sugar for (P says (Q says φ)) $\supset Q$ says φ. Notice that the definition of P reps Q on φ is a special case of controls and in effect asserts that P is a trusted authority with respect to Q saying φ.

$$\begin{aligned}
\mathcal{E}_{\mathcal{M}}[\![p]\!] &= I(p)\\
\mathcal{E}_{\mathcal{M}}[\![\neg\varphi]\!] &= W - \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \wedge \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!]\\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \vee \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!]\\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] &= (W - \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!]) \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!]\\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \equiv \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2 \supset \varphi_1]\!]\\
\mathcal{E}_{\mathcal{M}}[\![P \Rightarrow Q]\!] &= \begin{cases} W, & \text{if } J(Q) \subseteq J(P)\\ \emptyset, & \text{otherwise}\end{cases}\\
\mathcal{E}_{\mathcal{M}}[\![P \text{ says } \varphi]\!] &= \{w \mid J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\}\\
\mathcal{E}_{\mathcal{M}}[\![P \text{ controls } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![(P \text{ says } \varphi) \supset \varphi]\!]\\
\mathcal{E}_{\mathcal{M}}[\![P \text{ reps } Q \text{ on } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![P \mid Q \text{ says } \varphi \supset Q \text{ says } \varphi]\!]
\end{aligned}$$

Fig. 2. Semantics

3.2 Semantics

Kripke structures define the semantics of formulas.

Definition 1. *A* Kripke structure $\mathcal{M}$ *is a three-tuple* $\langle W, I, J\rangle$, *where:*

- W *is a nonempty set, whose elements are called* worlds.
- I : ***PropVar*** $\rightarrow \mathcal{P}(W)$ *is an* interpretation *function that maps each propositional variable p to a set of worlds.*
- J : ***PName*** $\rightarrow \mathcal{P}(W \times W)$ *is a function that maps each principal name A to a relation on worlds (i.e., a subset of $W \times W$).*

We extend J to work over arbitrary *principal expressions* using set union and relational composition as follows:

$$J(P\&Q) = J(P) \cup J(Q)$$
$$J(P \mid Q) = J(P) \circ J(Q),$$

where

$$J(P) \circ J(Q) = \{(w_1, w_2) \mid \exists w'.(w_1, w') \in J(P) \text{ and } (w', w_2) \in J(Q)\}$$

Definition 2. *Each Kripke structure $\mathcal{M} = \langle W, I, J\rangle$ gives rise to a function*

$$\mathcal{E}_{\mathcal{M}}[\![-]\!] : \textbf{\textit{Form}} \rightarrow \mathcal{P}(W),$$

where $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ is the set of worlds in which φ is considered true. $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ is defined inductively on the structure of φ, as shown in Figure 2.

Note that, in the definition of $\mathcal{E}_{\mathcal{M}}[\![P \text{ says } \varphi]\!]$, $J(P)(w)$ is simply the image of world w under the relation $J(P)$.

3.3 Inference Rules

In practice, relying on the Kripke semantics alone to reason about policies, CONOPS, and behavior is inconvenient. Instead, inference rules are used to manipulate formulas in the logic. All logical rules must be sound to maintain consistency.

$$\mathit{Taut}\quad \frac{}{\varphi}\quad \text{if } \varphi \text{ is an instance of a prop-logic tautology}$$

$$\mathit{Modus\ Ponens}\quad \frac{\varphi \qquad \varphi \supset \varphi'}{\varphi'} \qquad\qquad \mathit{Says}\quad \frac{\varphi}{P \text{ says } \varphi}$$

$$\mathit{MP\ Says}\quad \frac{}{(P \text{ says } (\varphi \supset \varphi')) \supset (P \text{ says } \varphi \supset P \text{ says } \varphi')}$$

$$\mathit{Speaks\ For}\quad \frac{}{P \Rightarrow Q \supset (P \text{ says } \varphi \supset Q \text{ says } \varphi)}$$

$$\mathit{Quoting}\quad \frac{}{P \mid Q \text{ says } \varphi \equiv P \text{ says } Q \text{ says } \varphi}$$

$$\mathit{\&Says}\quad \frac{}{P\&Q \text{ says } \varphi \equiv P \text{ says } \varphi \wedge Q \text{ says } \varphi}$$

$$\mathit{Idempotency\ of} \Rightarrow\quad \frac{}{P \Rightarrow P} \qquad\qquad \mathit{Monotonicity\ of} \mid\quad \frac{P' \Rightarrow P \qquad Q' \Rightarrow Q}{P' \mid Q' \Rightarrow P \mid Q}$$

$$\mathit{Associativity\ of} \mid\quad \frac{P \mid (Q \mid R) \text{ says } \varphi}{(P \mid Q) \mid R \text{ says } \varphi}$$

$$P \text{ controls } \varphi \quad \stackrel{\text{def}}{=} \quad (P \text{ says } \varphi) \supset \varphi$$

$$P \ \text{ reps } Q \text{ on } \varphi \stackrel{\text{def}}{=} P \mid Q \text{ says } \varphi \supset Q \text{ says } \varphi$$

Fig. 3. Core Inference Rules

$$\mathit{Quoting\ (1)}\quad \frac{P \mid Q \text{ says } \varphi}{P \text{ says } Q \text{ says } \varphi} \qquad\qquad \mathit{Quoting\ (2)}\quad \frac{P \text{ says } Q \text{ says } \varphi}{P \mid Q \text{ says } \varphi}$$

$$\mathit{Controls}\quad \frac{P \text{ controls } \varphi \qquad P \text{ says } \varphi}{\varphi} \qquad\qquad \mathit{Derived\ Speaks\ For}\quad \frac{P \Rightarrow Q \qquad P \text{ says } \varphi}{Q \text{ says } \varphi}$$

$$\mathit{Reps}\quad \frac{Q \text{ controls } \varphi \qquad P \ \text{ reps } Q \text{ on } \varphi \qquad P \mid Q \text{ says } \varphi}{\varphi}$$

$$\mathit{Rep\ Says}\quad \frac{P \ \text{ reps } Q \text{ on } \varphi \qquad P \mid Q \text{ says } \varphi}{Q \text{ says } \varphi}$$

Fig. 4. Derived Rules Used in this Paper

Definition 3. *A rule of form* $\dfrac{H_1 \cdots H_n}{C}$ *is sound if, for all Kripke structures* $\mathcal{M} = \langle W, I, J\rangle$*, if* $\mathcal{E}_{\mathcal{M}}[\![H_i]\!] = W$ *for each* $i \in \{1, \ldots, n\}$*, then* $\mathcal{E}_{\mathcal{M}}[\![C]\!] = W$*.*

The rules in Figures 3 and 4 are all sound. As an additional check, the logic and rules have been implemented in the HOL-4 (Higher Order Logic) theorem prover as a conservative extension of the HOL logic [2].

3.4 Confidentiality and Integrity Policies

Confidentiality and integrity policies such as Bell-LaPadula [3] and Biba's Strict Integrity policy [4], depend on classifying, i.e., assigning a confidentiality or integrity level to information, subjects, and objects. It is straightforward to extend the access-control logic to include confidentiality, integrity, or availability levels as needed. In what follows, we show how the syntax and semantics of *integrity* levels are added to the core access-control logic. The same process is used for levels used for *confidentiality* and *availability.*

Syntax. The first step is to introduce syntax for describing and comparing security levels. **IntLabel** is the collection of *simple integrity labels*, which are used as names for the integrity levels (e.g., HI and LO).

Often, we refer abstractly to a principal P's integrity level. We define the larger set **IntLevel** of *all* possible integrity-level expressions:

$$\textbf{IntLevel} ::= \textbf{IntLabel} \; / \; \mathsf{ilev}(\textbf{PName}).$$

A integrity-level expression is either a simple integrity label or an expression of the form $\mathsf{ilev}(A)$, where A is a simple principal name. Informally, $\mathsf{ilev}(A)$ refers to the integrity level of principal A.

Finally, we extend our definition of well-formed formulas to support comparisons of integrity levels:

$$\textbf{Form} ::= \textbf{IntLevel} \leq_i \textbf{IntLevel} \; / \; \textbf{IntLevel} =_i \textbf{IntLevel}$$

Informally, a formula such as $\textsc{lo} \leq_i \mathsf{ilev}(\mathit{Kate})$ states that Kate's integrity level is greater than or equal to the integrity level LO. Similarly, a formula such as $\mathsf{ilev}(\mathit{Barry}) =_i \mathsf{ilev}(\mathit{Joe})$ states that Barry and Joe have been assigned the same integrity level.

Semantics. Providing formal and precise meanings for the newly added syntax requires us to first extend our Kripke structures with additional components that describe integrity classification levels. Specifically, we introduce extended Kripke structures of the form

$$\mathcal{M} = \langle W, I, J, K, L, \preceq \rangle,$$

where:

- W, I, and J are as defined earlier.
- K is a non-empty set, which serves as the universe of *integrity levels*.
- $L : (\textbf{IntLabel} \cup \textbf{PName}) \rightarrow K$ is a function that maps each integrity label and each simple principal name to a integrity level. L is extended to work over arbitrary integrity-level expressions, as follows:
$$L(\mathsf{ilev}(A)) = L(A),$$
for every simple principal name A.
- $\preceq \subseteq K \times K$ is a partial order on K: that is, $\preceq$ is *reflexive* (for all $k \in K$, $k \preceq k$), *transitive* (for all $k_1, k_2, k_3 \in K$, if $k_1 \preceq k_2$ and $k_2 \preceq k_3$, then $k_1 \preceq k_3$), and *anti-symmetric* (for all $k_1, k_2 \in K$, if $k_1 \preceq k_2$ and $k_2 \preceq k_1$, then $k_1 = k_2$).

Using these extended Kripke structures, we extend the semantics for our new well-formed expressions as follows:

$$\mathcal{E}_{\mathcal{M}}[\![\ell_1 \leq_i \ell_2]\!] = \begin{cases} W, & \text{if } L(\ell_1) \preceq L(\ell_2) \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\mathcal{E}_{\mathcal{M}}[\![\ell_1 =_i \ell_2]\!] = \mathcal{E}_{\mathcal{M}}[\![\ell_1 \leq_i \ell_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\ell_2 \leq_i \ell_1]\!].$$

As these definitions suggest, the expression $\ell_1 =_i \ell_2$ is simply syntactic sugar for $(\ell_1 \leq_i \ell_2) \wedge (\ell_2 \leq_i \ell_1)$.

$$\ell_1 =_i \ell_2 \stackrel{\text{def}}{=} (\ell_1 \leq_i \ell_2) \wedge (\ell_2 \leq_i \ell_1)$$

$$\textit{Reflexivity of} \leq_i \quad \frac{}{\ell \leq_i \ell}$$

$$\textit{Transitivity of} \leq_i \quad \frac{\ell_1 \leq_i \ell_2 \qquad \ell_2 \leq_i \ell_3}{\ell_1 \leq_i \ell_3}$$

$$sl \leq_i \quad \frac{\mathsf{ilev}(P) =_i \ell_1 \quad \mathsf{ilev}(Q) =_i \ell_i \quad \ell_1 \leq_i \ell_2}{\mathsf{ilev}(P) \leq_i \mathsf{ilev}(Q)}$$

Fig. 5. Inference rules for relating integrity levels

Logical Rules. Based on the extended Kripke semantics we introduce logical rules that support the use of integrity levels to reason about access requests. Specifically, the definition, reflexivity, and transitivity rules in Figure 5 reflect that $\leq_i$ is a partial order. The fourth rule is derived and convenient to have.

4 Expressing Policy Elements in the Logic

With the definition of the syntax and semantics of access-control logic, we provide an introduction to expressing key elements of policy.

Statements and requests. Statements and requests are made by principals. Requests are logical statements. For example, if Alice wants to read file *foo*, we represent Alice's request as $Alice$ says $\langle read, foo \rangle$. We interpret $\langle read, foo \rangle$ as "it would be advisable to read file *foo*."

Credentials or certificates are statements, usually signed with a cryptographic key. For example, assume we believe public key K_{CA} is the key used by certificate authority CA. With this belief, we would interpret a statement made by K_{CA} to come from CA. In particular, if K_{CA} says $(K_{Alice} \Rightarrow Alice)$, we would interpret this public key certificate signed by K_{CA} as having come from CA.

Jurisdiction. Jurisdiction statements identify who or what has authority, specific privileges, powers, or rights. In the logic, jurisdiction statements usually are controls statements. For example, if Alice has the right to read file *foo*, we say $Alice$ controls $\langle read, foo \rangle$. If Alice has read jurisdiction on *foo* and Alice requests to read *foo*, then the *Controls* inference rule in Figure 4 allows us to infer $\langle read, foo \rangle$ is a sound decision, i.e.,

$$\frac{Alice \ \mathsf{controls}\ \langle read, foo \rangle \qquad Alice \ \mathsf{says}\ \langle read, foo \rangle}{\langle read, foo \rangle}.$$

Controls statements are also statements of trust. Suppose CA is recognized as the trusted authority on public-key certificates. If CA says $(K_{Alice} \Rightarrow Alice)$ then we believe that K_{Alice} is Alice's public key. An important consideration is that trust is not all or nothing in our logic. A principal may be trusted on some things but not others. For example, we may trust CA on matters related to Alice's key, but we may not trust CA on saying whether Alice has write permission on file *foo*. Essentially, the scope of trust of a principal is limited to the specific statements over which a principal has control.

Proxies and delegates. Often, principals who are the sources of requests or statements, do not in fact make the statements or requests themselves to the guards protecting a resource. Instead, something or somebody makes the request on their behalf. For example, it is quite common for cryptographic keys to be used as proxies, or stand-ins, for principals. In the case of certificate authority CA, we would say $K_{CA} \Rightarrow CA$. If we get a certificate signed using K_{CA}, then we would attribute the information in that certificate to CA. For example, using the *Derived Speaks For* rule in Figure 4 we can conclude that certificate authority CA vouches for K_{Alice} being Alice's public key:

$$\frac{K_{CA} \Rightarrow CA \qquad K_{CA} \textsf{ says } (K_{Alice} \Rightarrow Alice)}{CA \textsf{ says } (K_{Alice} \Rightarrow Alice).}$$

In situations where delegates are relaying orders or statements from their superiors, we typically use $\textsf{reps}$ formulas. For example, say Alice is Bob's delegate on withdrawing funds from $account_1$ and depositing funds into $account_2$. If we recognize Alice as Bob's delegate, we would write:

$$Alice \textsf{ reps } Bob \textsf{ on } (\langle withdraw\ \$10^6, account_1\rangle \wedge \langle deposit\ \$10^6, account_2\rangle).$$

From the semantics of $\textsf{reps}$, if we recognize Alice as Bob's delegate, in effect we are saying that Alice is trusted on Bob stating that he wishes a million dollars to be withdrawn from $account_1$ and deposited into $account_2$. If Alice says Bob says withdraw a million dollars from $account_1$ and deposit it into $account_2$, we will conclude that Bob has made the request. Using the *Rep Says* rule in Figure 4 we can conclude:

$$\frac{\begin{array}{c} Alice \textsf{ reps } Bob \textsf{ on } (\langle withdraw\ \$10^6, account_1\rangle \wedge \langle deposit\ \$10^6, account_2\rangle) \\ Alice \mid Bob \textsf{ says } (\langle withdraw\ \$10^6, account_1\rangle \wedge \langle deposit\ \$10^6, account_2\rangle) \end{array}}{Bob \textsf{ says } (\langle withdraw\ \$10^6, account_1\rangle \wedge \langle deposit\ \$10^6, account_2\rangle).}$$

5 An Extended Example

In this section we describe a hypothetical example CONOPS for joint operations where Joint Terminal Air Controllers (JTACs) on the ground identify targets and request they be destroyed. Requests are relayed to a theater command authority (TCA) by controllers in Airborne Early Warning and Control (AEW&C) aircraft. If approved by commanders, AEW&C controllers direct aircraft to destroy the identified target. To avoid threats due to compromised communications and control, the CONOPS specifies the use of a *mission validation appliance* (MVA) to authenticate requests and orders. What follows is a more detailed informal description of the scenario followed by a formalization and analysis of the CONOPS.

5.1 Scenario Description

The sequence of requests and approvals is as follows:

1. At the squad level, Joint Terminal Air Controllers (JTACs) are authorized to request air strikes against enemy targets in real time.
2. Requests are relayed to theater command authorities (TCAs) by Airborne Early Warning and Control (AEW&C) controllers.
3. Requested air strikes are approved by TCAs. These commanders are geographically distant from the squad requesting an air strike.
4. Command and control is provided by AEW&C aircraft operating close to the squad requesting an air strike.

Threat Avoidance. For mission security and integrity, JTACs, AEW&C controllers, pilots, and TCAs use a *mission validation appliance* (MVA) to request, transmit, authenticate, and authorize air strikes. MVAs are envisioned to be used as follows:

1. JTACs will use MVAs to transmit air strike requests to AEW&C controllers.
2. AEW&C controllers use MVAs to (a) authenticate JTACs, and (b) pass along JTAC requests to TCAs.
3. TCAs use MVAs to (a) authenticate JTACs and AEW&C controllers, and (b) send air strike authorizations to AEW&C controllers.
4. AEW&C controllers use MVAs to transmit air strike orders to pilots.

Security and Integrity Requirements. The CONOPS for using MVAs must meet the following security and integrity requirements.

- All requests, commands, and approvals must be authenticated. *No voice communications will be used.* This includes at a minimum:
  - All personnel are to be authenticated into mission roles, i.e., joint terminal air controller (JTAC), airborne early warning and controller (AEW&C) controller, pilot, theater command authority (TCA) , and security officer (SO).
  - All communications, commands, and approvals are to be encrypted and signed for integrity.
- All aircraft pilots receive their directions from AEW&C controllers and can only act with the approval of the TCA.
- All keys, certificates, and delegations, i.e., the foundation for trust, must be protected from corruption during operations. Only personnel with proper integrity levels are allowed to establish or modify the foundation of trust.

5.2 An Example CONOPS

MVA Use Cases. We consider two use cases. The first use case shows how MVAs are used when an air strike is requested by a JTAC. The second use case shows how MVAs are used when a TCA orders an air strike. Figure 6 illustrates the flow of requests starting from Alice as JTAC, through Bob as Controller, resulting in an authenticated request to Carol as TCA. The process starts with Alice using her token $Token_{Alice}$ to authenticate herself and her request to the JTAC MVA.

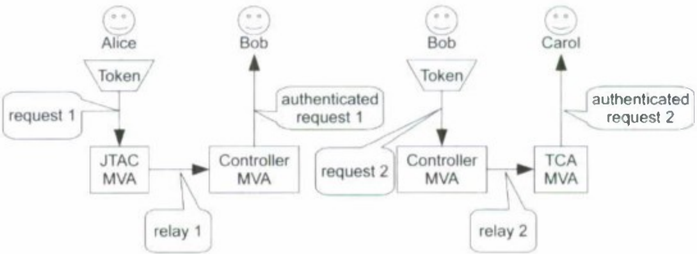


Fig. 6. Request Use Case

Table 1. Requests and Relayed Requests

Statement	**Formal Representation**
request 1	$(Token_{Alice} \mid JTAC)$ says $\langle strike, target \rangle$
relay 1	$(K_{JTAC\text{-}MVA} \mid JTAC)$ says $\langle strike, target \rangle$
authenticated request 1	$JTAC$ says $\langle strike, target \rangle$
request 2	$(Token_{Bob} \mid Controller)$ says $(JTAC$ says $\langle strike, target \rangle)$
relay 2	$(K_{Controller\text{-}MVA} \mid Controller)$ says $(JTAC$ says $\langle strike, target \rangle)$
authenticated request 2	$Controller$ says $(JTAC$ says $\langle strike, target \rangle)$

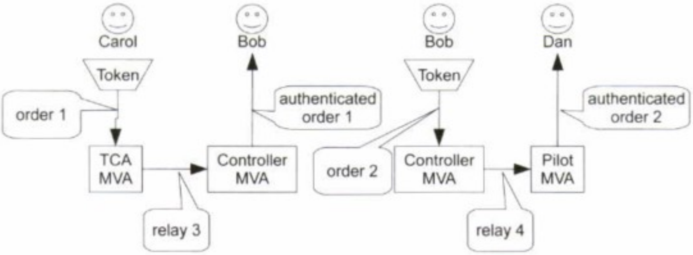


Fig. 7. Order Use Case

Table 2. Orders and Relayed Orders

Statement	**Formal Representation**
order 1	$(Token_{Carol} \mid TCA)$ says $\langle strike, target \rangle$
relay 3	$(K_{TCA-MVA} \mid TCA)$ says $\langle strike, target \rangle$
authenticated order 1	TCA says $\langle strike, target \rangle$
order 2	$(Token_{Bob} \mid Controller)$ says $(TCA$ says $\langle strike, target \rangle)$
relay 4	$(K_{Controller-MVA} \mid Controller)$ says $(TCA$ says $\langle strike, target \rangle)$
authenticated order 2	$Controller$ says $(TCA$ says $\langle strike, target \rangle)$

The JTAC MVA authenticates Alice and her role, and relays Alice's request using its key, $K_{JTAC\text{-}MVA}$ to the Controller MVA. The Controller MVA authenticates the JTAC MVA and presents the authenticated request to Bob.

Should Bob decide to pass on Alice's request, he uses his token to authenticate himself to the Controller MVA, which relays his request to the TCA MVA, which presents the authenticated request to Carol, a Theater Command Authority. Table 1 lists the formal representation of each request, relayed request, and authenticated request in Figure 6.

Figure 7 shows a similar flow of orders starting from Carol as TCA, through Bob as Controller, resulting in an authenticated order to Dan as Pilot. Carol authenticates herself to the TCA MCA using her token. Her orders are relayed to Bob. When Bob decides to pass on the order to Dan, he does so by authenticating himself to the Controller MVA, which relays to orders to Dan via the Pilot MVA. The formulation of each order and relayed order is shown in Table 2.

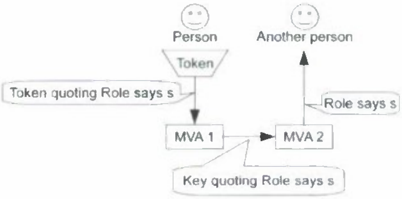


Fig. 8. General Pairing of MVAs

Table 3. Statements and Relayed Statements

Statement	Formal Representation
statement	$(Token \mid Role)$ says φ
relayed statement	$(K_{MVA-1} \mid Role)$ says φ
authenticated statement	$Role$ says φ

Deducing Policies, Certifications, Delegations, and Trust Assumptions. Based on the use cases for air strike requests and air strike orders, we determine what policies, certifications, delegations, and trust assumptions are required to justify each MVA action in the CONOPS. We look at each MVA's input and output, and based on the CONOPS, infer what policies, certifications, delegations, and trust assumptions are required. We look for repeated patterns of behavior that lead to repeated patterns of reasoning. Both use cases exhibit the same pattern of behavior as illustrated in Figure 8 and formulated in Table 3.

1. A person authenticates herself and claims a role using a token. Acting in a role, the person makes a statement (request or order). The first MVA, *MVA 1*, authenticates both the person and the role, and then relays the statement using its key to the second MVA, *MVA 2*.
2. *MVA 2* authenticates *MVA 1* and the role it is serving, then passes the statement up to the person using *MVA 2*.

Given the repeated pattern, we prove two *derived inference rules* (*MVA 1* and *MVA 2*) that justify the behavior of *MVA 1* and *MVA 2*.

$$MVA\ 1 \quad \frac{\begin{array}{c}(Token \mid Role) \text{ says } \varphi \\ K_{Auth} \text{ says } (Person \text{ reps } Role \text{ on } \varphi) \\ K_{Auth} \text{ says } (Token \Rightarrow Person) \\ Auth \text{ controls } (Person \text{ reps } Role \text{ on } \varphi) \\ Auth \text{ controls } (Token \Rightarrow Person) \\ K_{Auth} \Rightarrow Auth\end{array}}{K_{MVA_1} \mid Role \text{ says } \varphi}$$

$$MVA\ 2 \quad \frac{\begin{array}{c}(K_{MVA_1} \mid Role) \text{ says } \varphi \\ K_{Auth} \text{ says } (MVA_1 \text{ reps } Role \text{ on } \varphi) \\ K_{Auth} \text{ says } (K_{MVA_1} \Rightarrow MVA_1) \\ Auth \text{ controls } (MVA_1 \text{ reps } Role \text{ on } \varphi) \\ Auth \text{ controls } (K_{MVA_1} \Rightarrow MVA_1) \\ K_{Auth} \Rightarrow Auth\end{array}}{Role \text{ says } \varphi}$$

Both rules have the same components, as shown in Table 4. The components have the following functions:

Table 4. MVA Inputs, Outputs, Certificates, Jurisdiction, and Trust Assumptions

Item	Formula
Input	$(\mathit{Token\ or\ Key} \mid \mathit{Role})$ says φ
Delegation Certificate	K_{Auth} says $(\mathit{Person\ or\ Object}$ reps Role on $\varphi)$
Key Certificate	K_{Auth} says $(\mathit{Token\ or\ Key} \Rightarrow \mathit{Person\ or\ Object})$
Jurisdiction	Auth $\mathsf{controls}$ $(\mathit{Person\ or\ Object}$ reps Role on $\varphi)$
Jurisdiction	Auth $\mathsf{controls}$ $(\mathit{Token\ or\ Key} \Rightarrow \mathit{Person\ or\ Object})$
Trust Assumption	$K_{Auth} \Rightarrow \mathit{Auth}$

1. *input*: a token or key quoting a role
2. *certificate*: a certificate authorizing a delegation
3. *certificate*: a public key certificate
4. *jurisdiction*: an assumption about an authority's jurisdiction to authorize a person or MVA to act in a role
5. *jurisdiction*: an assumption about an authority's jurisdiction over keys
6. *trust assumption*: knowledge of the trusted authority's key

Both rules have nearly identical proofs that are direct application of inference rules described in Section 3.3.

Using the inference rule *MVA 1*, we easily prove the following rule for the *TCA MVA* authenticating Carol and validating her order for an air strike, where *SO* is the *Security Officer* role, the *SO* has jurisdiction over roles and keys, and K_S is the key that speaks for the *SO*.

$$\mathit{TCA\text{-}MVA}\quad \frac{\begin{array}{c} \mathit{Token}_{Carol} \mid \mathit{TCA}\ \mathsf{says}\ \langle \mathit{strike}, \mathit{target}\rangle \\ K_{SO}\ \mathsf{says}\ (\mathit{Carol}\ \ \mathsf{reps}\ \mathit{TCA}\ \mathsf{on}\ \langle \mathit{strike}, \mathit{target}\rangle) \\ K_{SO}\ \mathsf{says}\ \mathit{Token}_{Carol} \Rightarrow \mathit{Carol} \\ \mathit{SO}\ \mathsf{controls}\ \mathit{Token}_{Carol} \Rightarrow \mathit{Carol} \\ \mathit{SO}\ \mathsf{controls}\ (\mathit{Carol}\ \ \mathsf{reps}\ \mathit{TCA}\ \mathsf{on}\ \langle \mathit{strike}, \mathit{target}\rangle) \\ K_{SO} \Rightarrow \mathit{SO} \end{array}}{K_{\mathit{TCA\text{-}MVA}} \mid \mathit{TCA}\ \mathsf{says}\ \langle \mathit{strike}, \mathit{target}\rangle}$$

Similar rules and proofs are written for each MVA. The above discussion on certificates installed properly in MVAs leads us to the final use case, namely the *trust establishment* use case.

5.3 Trust Establishment

Biba's Strict Integrity model [4] is the basis for maintaining integrity of the MVAs. As Strict Integrity is the dual of Bell and LaPadula's confidentiality model [3], the short summary of Strict Integrity is, *no read down and no write up*. For subjects S and objects O, S may have discretionary read rights on O if O's integrity level meets or exceeds S's. For write access, S's integrity level must meet or exceed O's.

$$\begin{array}{l} \mathsf{ilev}(S) \leq_i \mathsf{ilev}(O) \supset S\ \mathsf{controls}\ \langle \mathit{read}, O\rangle \\ \mathsf{ilev}(O) \leq_i \mathsf{ilev}(S) \supset S\ \mathsf{controls}\ \langle \mathit{write}, O\rangle. \end{array}$$

There are two integrity levels: L_{op} and L_{Sec}, where $L_{op} \leq_i L_{Sec}$. All certificates have an integrity level L_{Sec}, i.e., $\mathsf{ilev}(\mathit{cert}) =_i L_{Sec}$. Table 5 show the integrity

Table 5. Roles and Rights to Certificates

Role	Rights
SO (L_{Sec})	install, read
JTAC (L_{op})	read
Controller (L_{op})	read
TCA (L_{op})	read
Pilot (L_{op})	read

level and certificate access rights for each role. Strict integrity is satisfied as only the security officer SO (with the same integrity level L_{Sec} as certificates) can install or write certificates into MVAs. Every other role is at the L_{op} level and can only read certificates.

Installing K_{SO}. Establishing the basis for trust in MVAs starts with the installation of the *Security Officer's* key, K_{SO}. This is assumed to be done by controlled physical access to each MVA that is deployed. Once the Security Officer's key is in place, the certificates that an MVA needs can be installed.

Certificate Installation. Suppose Erica is acting as the Security Officer SO. The policy is that security officers can install certificates, if the SO has a high enough integrity level, and is given by

$$\mathsf{ilev}(cert) \leq_i \mathsf{ilev}(SO) \supset SO \text{ controls } \langle install, cert \rangle .$$

Erica's authorization to act in the Security Officer role to install certificates is given by

$$K_{so} \text{ says } Erica \text{ reps } SO \text{ on } \langle install, cert \rangle .$$

This authorization is accepted under the assumption that $K_{SO} \Rightarrow SO$ and that the SO has jurisdiction, which is given by

$$SO \text{ controls } Erica \text{ reps } SO \text{ on } \langle install, cert \rangle .$$

The proof for justifying Erica's capability to install certificates acting as a Security Officer, assuming her integrity level is L_{so} is a straightforward application of inference rules described in Section 3.3.

6 Related Work

The access-control logic we use is based on Abadi and Plotkin's work [5], with modifications described in [6]. Many other logical systems have been used to reason about access control. Some of them are summarized in [7].

Our contribution is the methodology and application of logic to describe policies, operations, and assumptions in CONOPS. Moreover, we have implemented this logic in the HOL-4 theorem prover, which provides both an independent verification of soundness as well as support for computer-assisted reasoning.

7 Conclusions

Our objective is the put usable mathematical methods into the hands of practicing engineers to help them reason about policies and concepts of operations. We have experimented with policy-based design and verification for five years in the US Air Force's Advanced Course in Engineering (ACE) Cybersecurity Bootcamps [8]. Our experience with a wide variety of students, practicing engineers, and Air Force officers suggests that using the access-control logic meets this objective.

References

1. Coram, R.: Boyd: The Fighter Pilot who Changed the Art of War. Back Bay Books/Little, Brown and Company (2002)
2. Gordon, M., Melham, T.: Introduction to HOL: A Theorem Proving Environment for Higher Order Logic. Cambridge University Press, New York (1993)
3. Bell, D.E., La Padula, L.J.: Secure computer systems: Mathematical foundations. Technical Report Technical Report MTR-2547, Vol. 1, MITRE Corporation, Bedford, MA (March 1973)
4. Biba, K.: Integrity considerations for secure computer systems. Technical Report MTR-3153, MITRE Corporation, Bedford, MA (June 1975)
5. Abadi, M., Burrows, M., Lampson, B., Plotkin, G.: A Calculus for Access Control in Distributed Systems. ACM Transactions on Programming Languages and Systems 15(4), 706–734 (1993)
6. Chin, S.K., Older, S.: Reasoning about delegation and account access in retail payment systems. In: MMM-ACNS (2007)
7. Abadi, M.: Logic in access control (tutorial notes), 145–165 (2009)
8. Chin, S.K., Older, S.: A rigorous approach to teaching access control. In: Proceedings of the First Annual Conference on Education in Information Security. ACM, New York (2006)

Using Equivalence Relations for Corrective Enforcement of Security Policies

Raphaël Khoury and Nadia Tawbi

Department of Computer Science and Software Engineering, Laval University,
1065, avenue de la Médecine, Québec (QC), Canada G1V 0A6
raphael.khoury.1@ulaval.ca, Nadia.Tawbi@ift.ulaval.ca

Abstract. In this paper, we present a new framework of runtime security policy enforcement. Building on previous studies, we examine the enforcement power of monitors able to transform their target's execution, rather than simply accepting it if it is valid, or aborting it otherwise. We bound this ability by a restriction stating that any transformation must preserve equivalence between the monitor's input and output. We proceed by giving examples of meaningful equivalence relations and identify the security policies that are enforceable with their use. We also relate our work to previous findings in this field. Finally, we investigate how an a priori knowledge of the target program's behavior would increase the monitor's enforcement power.

Keywords: Monitoring, Security Policy Enforcement, Program Transformation, inlined reference monitors.

1 Introduction

In light of the increasing complexity and interconnectivity of modern software, there is a growing realization that formal security frameworks are needed to ensure code safety. Because they have solid theoretical underpinnings, such framework can provide assurance that the desired security policy will be enforced regardless of the target program's output. One such formal security framework, which has gained wide acceptance in recent years is runtime monitoring. This approach to code safety seeks to allow an untrusted code to run safely by observing its execution and reacting if need be to prevent a potential violation of a user-supplied security policy.

The monitor is modeled as an automaton which takes the program's execution as input, and outputs an alternate execution, usually by truncating the input if it is invalid. Several studies have focused on establishing the set of security policies that are enforceable by monitors operating under various constraints. This is necessary to best select the appropriate enforcement mechanism given the desired security policy and enforcement context. In this study, we take this framework one step further and examine the enforcement power of monitors capable of transforming their input. However, the monitor's ability to do so must be constrained by a requirement to maintain an equivalence between input

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 139–154, 2010.

and output. This intuitively corresponds to an enforcement paradigm, closer to one that would be encountered in practice, in which the actions taken by the monitor are constrained by a limitation that certain behaviors present in the original sequence be preserved.

The question of identifying the set of security policies (termed properties) enforceable by monitors able to transform invalid executions was raised several times in the literature [16,4,13,10]. While these studies observe that this ability considerably extends the monitor's enforcement power, they do not provide a more specific characterization of the set of enforceable properties w.r.t equivalence relations other than syntactic equality. This results from the lack of a framework constraining the ability of a monitor to transform its input. This point is concisely explained by Ligatti et al. in [13]. "*A major difficulty with semantic equivalence is its generality: for any reasonable property $\hat{\mathcal{P}}$ there exists a sufficiently helpful equivalence relation that enables a security automaton to enforce $\hat{\mathcal{P}}$*".

Indeed, the authors go on to note that if all valid sequences can be thought of as being equivalent to one another, any security policy can be enforced simply by always outputting the same valid arbitrarily chosen sequence for all inputs. This strictly meets the definition of enforcement but does not provide a meaningful enforcement of the desired policy.

For example, consider a system managing online purchases, and a security policy forbidding a user from browsing certain merchandise without prepaying. A monitor could abort the execution as soon as this is attempted. But the property would also be enforced by replacing the input sequence with any sequence of actions respecting the policy, even if it contains purchases unrequested by any users, or by outputting nothing, depriving legitimate users of the ability to use the system.

In this paper, we suggest a framework to study the enforcement power of monitors. The key insight behind our work is to state certain criteria which must be met for an equivalence relation to be useful in monitoring. We then give two examples of such equivalence relations, and show which security properties are enforceable with their use.

The contributions of this paper are as follows: First, we develop a framework of enforcement, termed corrective$_{\cong}$ enforcement to reason about the enforcement power of monitors bounded to produce an output which is semantically equivalent to their input with respect to some equivalence relation $\cong$. We suggest two possible examples of such relations and give the set of enforceable security policies as well as examples of real policies for each. Finally, we show that the set of enforceable properties defined in [13] for effective enforcement can be considered as special cases of our more general framework.

The remainder of this paper is organized as follows. Section 2 presents a review of related work. In Section 3, we define some concepts and notations that are used throughout the paper. In Section 4, we show under what conditions equivalence relations can be used to transform sequences and ensure the respect of the security policy. The set of security policies which can be enforced in this manner is examined in Section 5. In Section 6, we give two examples of possible equivalence relations and show that they can serve as the basis for the enforcement of meaningful security properties. In section 7, we investigate how an a priori knowledge of the target program's behavior would increase the monitor's enforcement power. Concluding remarks and avenues for future work are laid out in Section 8.

2 Related Work

Schneider, in his seminal work [16], was the first to investigate the question of which security policies could be enforced by monitors. He focused on specific classes of monitors, which observe the execution of a target program with no knowledge of its possible future behavior and with no ability to affect it, except by aborting the execution. Under these conditions, he found that a monitor could enforce the precise security policies that are identified in the literature as *safety* properties, and are informally characterized by prohibiting a certain bad thing from occurring in a given execution.

Schneider's study also suggested that the set of properties enforceable by monitors could be extended under certain conditions. Building on this insight, Ligatti, Bauer and Walker [4,12] examined the way the set of policies enforceable by monitors would be extended if the monitor had some knowledge of its target's possible behavior or if its ability to alter that behavior were increased. The authors modified the above definition of a monitor along three axes, namely (1) the means at the disposal of the monitor in order to respond to a possible violation of the security policy; (2) whether the monitor has access to information about the program's possible behavior; and (3) how strictly the monitor is required to enforce the security policy. Consequently, they were able to provide a rich taxonomy of classes of security policies, associated with the appropriate model needed to enforce them. Several of these models are strictly more powerful than the security automata developed by Schneider and are used in practice.

Evolving along this line of inquiry, Ligatti et al. [13] gave a more precise definition of the set of properties enforceable by the most powerful monitors, while Fong [9] and Talhi et al. [18] expounded on the capabilities of monitors operating under memory constraints. Hamlen et al. [10] , on the other hand, showed that in-lined monitors (whose operation is injected into the target program's code, rather than working in parallel) can also enforce more properties than those modeled by a security automaton. In [3], a method is given to enforce both safety and *co-safety* properties by monitoring. The set of properties enforceable by monitors aided by static analysis of the program is examined in [6,7]. In [5], Bielova et al. delineate the set of properties enforceable by a monitor limited to suppressing a finite subsequence of the execution before either outputting or deleting them. In [15], Ligatti et al. propose an alternate, more general model of monitoring, which imposes on the monitor that it respond to the target program's actions in lock step.

3 Preliminaries

Let us briefly start with some preliminary definitions.

Executions are modeled as sequences of atomic actions taken from a finite or countably infinite set of actions Σ. The empty sequence is noted ϵ, the set of all finite length sequences is noted Σ^*, that of all infinite length sequences is noted Σ^ω, and the set of all possible seqenees is noted $\Sigma^\infty = \Sigma^\omega \cup \Sigma^*$. Likewise, for a set of sequences $\mathcal{S}$, $\mathcal{S}^*$ denote the finite iterations of sequences of $\mathcal{S}$ and $\mathcal{S}^\omega$ that of infinite iterations, and $\mathcal{S}^\infty = \mathcal{S}^\omega \cup \mathcal{S}^*$. Let $\tau \in \Sigma^*$ and $\sigma \in \Sigma^\infty$ be two sequences of actions. We write $\tau;\sigma$ for the concatenation of τ and σ. We say that τ is a prefix of σ noted $\tau \preceq \sigma$, or equivalently $\sigma \succeq \tau$ *iff* there exists a sequence σ' such that $\tau;\sigma' = \sigma$. We write $\tau \prec \sigma$ (resp. $\sigma \succ \tau$) for $\tau \preceq \sigma \wedge \tau \neq \sigma$ (resp. $\sigma \succ \tau \wedge \tau \neq \sigma$). Finally, let $\tau, \sigma \in \Sigma^\infty$, τ is said to be a suffix of σ iff there exists a $\sigma' \in \Sigma^*$ s.t. $\sigma = \sigma';\tau$.

We denote by $pref(\sigma)$ (resp. $suf(\sigma)$) the set of all prefixes (resp. suffixes) of σ. Let $A \subseteq \Sigma^\infty$ be a subset of sequences. Abusing the notation, we let $pref(A)$ (resp. $suf(A)$)stands for $\bigcup_{\sigma \in A} pref(\sigma)$ (resp. $\bigcup_{\sigma \in A} suf(\sigma)$). The i^{th} action in a sequence σ is given as σ_i, σ_1 denotes the first action of σ, $\sigma[i,j]$ denotes the sequence occurring between the i^{th} and j^{th} actions of σ, and $\sigma[i,..]$ denotes the remainder of the sequence, starting from action σ_i. The length of a sequence $\tau \in \Sigma^*$ is given as $|\tau|$.

A multiset, or bag [17] is a generalization of a set in which each element may occur multiple times. A multiset $\mathcal{A}$ can be formally defined as a pair $\langle A, f\rangle$ where A is a set and $f : A \rightarrow \mathbb{N}$ is a function indicating the number of occurrences of each element of A in $\mathcal{A}$. Note that $a \notin A \Leftrightarrow f(a) = 0$. Thus, by using this insight, to define basic operations on multisets one can consider a universal set A and different functions of type $A \rightarrow \mathbb{N}$ associated with it to form different multisets.

Given two multisets $\mathcal{A} = \langle A, f\rangle$ and $\mathcal{B} = \langle A, g\rangle$,the multiset union $\mathcal{A} \cup \mathcal{B} = \langle A, h\rangle$ where $\forall a \in A : h(a) = f(a) + g(a)$. Furthermore, $\mathcal{A} \subseteq \mathcal{B} \Leftrightarrow \forall a \in A : f(a) \leq g(a)$. The removal of an element $a \in A$ from multiset $\mathcal{A}$ is done by updating the function f so that $f(a) = max(f(a) - 1, 0)$.

Finally, a security policy $P \subseteq \Sigma^\infty$ is a set of allowed executions. A policy P is a property iff there exists a decidable predicate $\hat{\mathcal{P}}$ over the executions of Σ^∞ s.t. $\sigma \in P \Leftrightarrow \hat{\mathcal{P}}(\sigma)$. In other words, a property is a policy for which the membership of any sequence can be determined by examining only the sequence itself. Such a sequence is said to be valid or to respect the property. Since all policies enforceable by monitors are properties, we use $\hat{\mathcal{P}}$ to refer to policies and their characteristic predicate interchangeably. Properties for which the empty sequence ϵ is a member are said to be *reasonable*.

A number of classes of properties have been defined in the literature and are of special interest in the study of monitoring. First are safety properties [11], which proscribe that certain "bad things" occur during the execution. Let Σ be a set of actions and $\hat{\mathcal{P}}$ be a property, $\hat{\mathcal{P}}$ is a safety property iff

$$\forall \sigma \in \Sigma^\infty : \neg\hat{\mathcal{P}}(\sigma) \Rightarrow \exists \sigma' \preceq \sigma : \forall \tau \succeq \sigma' : \neg\hat{\mathcal{P}}(\tau) \qquad \text{(safety)}$$

Alternatively, a *liveness* property [2] is a property prescribing that a certain "good thing" must occur in any valid execution. Formally, for an action set Σ^∞ and a property $\hat{\mathcal{P}}$, $\hat{\mathcal{P}}$ is a liveness property iff

$$\forall \sigma \in \Sigma^* : \exists \tau \in \Sigma^\infty : \tau \succeq \sigma \wedge \hat{\mathcal{P}}(\tau) \qquad \text{(liveness)}$$

Any property can be stated as the conjunction of a safety property and a liveness property [1]. Another relevant set of properties is that of infinite renewal properties (*renewal*), defined in [13] to characterize the set properties enforceable by edit-automata monitors using syntactic equality as the equivalence relation. A property is member of this set if every infinite valid sequence has infinitely many valid prefixes, while every invalid infinite sequence has only finitely many such prefixes. Formally, for an action set Σ and a property $\hat{\mathcal{P}}$, $\hat{\mathcal{P}}$ is a renewal property iff it meets the following two equivalent conditions

$$\forall \sigma \in \Sigma^\omega : \hat{\mathcal{P}}(\sigma) \Leftrightarrow \{\sigma' \preceq \sigma | \hat{\mathcal{P}}(\sigma')\} \textit{is an infinite set} \qquad \text{(renewal}_1\text{)}$$

$$\forall \sigma \in \Sigma^\omega : \hat{\mathcal{P}}(\sigma) \Leftrightarrow (\forall \sigma' \preceq \sigma : \exists \tau \preceq \sigma : \sigma' \preceq \tau \wedge \hat{\mathcal{P}}(\tau)) \qquad \text{(renewal}_2\text{)}$$

Note that the definition of renewal imposes no restrictions on the finite sequences in $\hat{\mathcal{P}}$. For infinite sequences, the set of renewal properties includes all safety properties, some liveness properties and some properties which are neither safety nor liveness.

Finally, we formalize the the set of *transactional* properties, suggested in [13], which will be of use in section 6.1. A transactional property is one in which any valid sequence consists of a concatenation of valid finite transactions. Such properties can model, for example, the behavior of systems which repeatedly interacts with clients using a well defined protocol, such as a system managing the allocation of resource or the access to a database. Let Σ be an action set and $T \subseteq \Sigma^*$ be a subset of finite transactions. $\hat{\mathcal{P}}_T$ is a transactional property over set T iff

$$\forall \sigma \in \Sigma^\infty : \hat{\mathcal{P}}_T(\sigma) \Leftrightarrow \sigma \in T^\infty \qquad \text{(transactional)}$$

This definition is subtly different, and indeed forms a subset, to that of iterative properties defined in [5]. transactional properties also form a subset to the set of renewal properties, and include some but not all safety properties, liveness properties as well as properties which are neither.

4 Monitoring with Equivalence Relations

The idea of using equivalence relations to transform execution sequences was first suggested in [10]. The equivalence relations are restricted to those that are *consistent* with the security policy under consideration. Let $\hat{\mathcal{P}}$ be a security policy, the consistency criterion for an equivalence relation $\cong$ is given as:

$$\forall \sigma, \sigma' \in \Sigma^\infty : \sigma \cong \sigma' \Rightarrow \hat{\mathcal{P}}(\sigma) \Leftrightarrow \hat{\mathcal{P}}(\sigma'). \qquad \text{(consistency)}$$

Yet, upon closer examination, this criterion seems too restrictive for our purposes. If any two equivalent sequences always meet this criterion, an invalid prefix can never be made valid by replacing it with another equivalent one. It is thus impossible to "correct" an invalid prefix and output it.

It is still necessary to impose some restrictions on equivalence relations and their relation to properties. Otherwise, as discussed above, any property would be enforceable, but not always in a meaningful manner.

In this paper, we suggest the following alternative framework.

Following previous work in monitoring by Fong [9], we use an abstraction function $\mathcal{F} : \Sigma^* \rightarrow \mathcal{I}$, to capture the property of the input sequence which the monitor must preserve throughout its manipulation. While Fong made use of abstractions to reduce the overhead of the monitor, we use them as the basis for our equivalence relations. Such an abstraction can capture any property of relevance. This may be, for example, the presence of certain subwords or factors or any other semantic property of interest. We expect the property to be consistent with this abstraction rather than with the equivalence relation itself. Formally:

$$\mathcal{F}(\sigma) = \mathcal{F}(\sigma') \Rightarrow \hat{\mathcal{P}}(\sigma) \Leftrightarrow \hat{\mathcal{P}}(\sigma') \tag{4.1}$$

Furthermore, we restrict ourselves to equivalence relations which group together sequences for which the abstraction is similar. To this end, we let $\leq$ stand for some partial order over the values of $\mathcal{I}$. We define $\sqsubseteq$ as the partial order defined as $\forall \sigma, \sigma' \in \Sigma^* : \sigma \sqsubseteq \sigma' \Leftrightarrow \mathcal{F}(\sigma) \leq \mathcal{F}(\sigma')$. We equivalently write $\sigma' \sqsupseteq \sigma$ and $\sigma \sqsubseteq \sigma'$.

The transformation performed by the monitor on a given sequence τ produces a new sequence τ' s.t. $\tau' \sqsubseteq \tau$. To ease the monitor's task in finding such a suitable replacement, we impose the following two constraints on the equivalence relations used in monitoring.

First, if two sequences are equivalent, any intermediary sequence over $\sqsubseteq$ is also equivalent to them.

$$\sigma \sqsubseteq \sigma' \sqsubseteq \sigma'' \wedge \sigma \cong \sigma'' \Rightarrow \sigma \cong \sigma' \tag{4.2}$$

Second, two sequences cannot be equivalent if they do not share a common greatest lower bound.Conversely, the greatest lower bound of two equivalent sequences is also equivalent to them. These last two criteria are stated together as:

$$\forall \sigma, \sigma' \in \Sigma^* : \sigma \cong \sigma' \Rightarrow \exists \tau \in \Sigma^* : \tau = (\sigma \sqcap \sigma') \wedge \tau \cong \sigma \tag{4.3}$$

where $(\sigma \sqcap \sigma') = \tau$ s.t. $\tau \sqsubseteq \sigma \wedge \tau \sqsubseteq \sigma' \wedge \neg(\exists \tau' \sqsupseteq \tau : \tau' \sqsubseteq \sigma \wedge \tau' \sqsubseteq \sigma')$

The intuition behind the above two restrictions, is that, if an equivalence restriction meets these two criteria, a monitor looking for a valid sequence equivalent to an invalid input simply has to iteratively perform certain transformations until such a sequence is found or until every equivalent sequence has been examined.

We define our equivalence relations over finite sequences. Two infinite sequences are equivalent, iff they have infinitely many valid equivalent prefixes.

Let $\cong$ be an equivalence relation over the sequences of Σ^*

$$\forall \sigma, \sigma' \in \Sigma^\omega : \sigma \cong \sigma' \Leftrightarrow \forall \tau \prec \sigma : \exists \upsilon \succeq \tau : \exists \tau' \prec \sigma' : \upsilon \cong \tau' \tag{4.4}$$

It is easy to see that an equivalence between infinite sequence not meeting this criterion would be of no use to a monitor, which is bound to transform its input in finite time.

Finally, we impose the following closure restriction:

$$\tau \cong \tau' \Rightarrow \tau;\sigma \cong \tau';\sigma \tag{4.5}$$

This may, at first sight, seem like an extremely restrictive condition to be imposed but in fact every meaningful relation that we examined has this property.

Furthermore, no security property can be enforced using an equivalence relation lacking this property. Consider for example what would happen if a monitor is presented with an invalid prefix τ of a longer input sequence for which there exists a valid equivalent sequence τ'. It would be natural for the monitor to transform τ into τ'. Yet it would also be possible that the full original sequence $\sigma \succ \tau$ be actually valid, but that there exists no equivalent sequence for which τ' is a prefix.

In fact, $\sqsubseteq$ organizes the sequences according to some semantic framework, using values given by an abstraction function $\mathcal{F}$, $\hat{\mathcal{P}}$ establishes that only certain values of $\mathcal{F}$ are valid or that a certain threshold must be reached, while $\cong$ groups the sequences if their abstractions are equivalent. In section 6, we give examples that show how the framework described in this section can be used to model desirable security properties of programs and meaningful equivalence relations between their executions.

5 Corrective Enforcement

In this section, we present the automata-based model used to study the enforcement mechanism, and give a more formal definition of our notion of enforcement.

The edit automaton [4,13] is the most general model of a monitor. It captures the behavior of a monitor capable of inserting or suppressing any action, as well as halting the execution in progress.

Definition 1. *An edit automaton is a tuple $\langle \Sigma, Q, q_0, \delta \rangle$ where*[1]*:*

- *Σ is a finite or countably infinite set of actions;*
- *Q is a finite or countably infinite set of states;*
- *$q_0 \in Q$ is the initial state;*
- *$\delta : (Q \times \Sigma) \rightarrow (Q \times \Sigma^\infty)$ is the transition function, which, given the current state and input action, specifies the automaton's output and successor*

[1] This definition, taken from [18], is equivalent to the one given in [4].

state. At any step, the automaton may accept the action and output it intact, suppress it and move on to the next action, output nothing, or output some other sequence in Σ^∞. If at a given state the transition for a given action is undefined, the automaton aborts.

Let $\mathcal{A}$ be an edit automaton, we let $\mathcal{A}(\sigma)$ be the output of $\mathcal{A}$ when its input is σ.

Most studies on this topic have focused on effective enforcement. A mechanism effectively enforces a security property iff it respects the two following principles, from [4]:

1. *Soundness* : All output must respect the desired property.
2. *Transparency* : The semantics of executions which already respect the property must be preserved. This naturally requires the use of an equivalence relation, stating when one sequence can be substitnted for another.

Definition 2. *Let $\mathcal{A}$ be an edit automaton. $\mathcal{A}$ effectively$_{\cong}$ enforces the property $\hat{\mathcal{P}}$ iff $\forall\sigma \in \Sigma^\infty$*

1. *$\hat{\mathcal{P}}(\mathcal{A}(\sigma))$ (i.e. $\mathcal{A}(\sigma)$ is valid)*
2. *$\hat{\mathcal{P}}(\sigma) \Rightarrow \mathcal{A}(\sigma) \cong \sigma$*

In the literature, the only equivalence relation $\cong$ for which the set of effectively$_{\cong}$ enforceable properties has been formally studied is syntactic equality[4]. Yet, effective enforcement is only one paradigm of enforcement which has been suggested. Other enforcement paradigms include precise enforcement[4], all-or-nothing delayed enforcement[5] or conservative enforcement[4].

In this study, we introduce a new paradigm of security property enforcement, termed corrective$_{\cong}$ enforcement. An enforcement mechanism correctively$_{\cong}$ enforces the desired property if every output sequence is both valid and equivalent to the input sequence. This captures the intuition that the monitor is both required to output a valid sequence, and forbidden from altering the semantics of the input sequence. Indeed, it is not always reasonable to accept, as do preceding studies of monitor's enforcement power, that the monitor is allowed to replace an invalid execution with any valid sequence, even ϵ. A more intuitive model of the desired behavior of a monitor would rather require that only minimal alterations be made to an invalid sequence, for instance by releasing a resource or adding an entry in a log. Those parts of the input sequence which are valid, should be preserved in the output, while invalid behaviors should be corrected or removed. It is precisely these corrective behaviors that we seek to model using our equivalence relations. The enforcement paradigm thus ensures that the output is always valid, and that all valid behavior intended by the user in the input, is present in the monitor's output.

Definition 3. *Let $\mathcal{A}$ be an edit automaton. $\mathcal{A}$ correctively$_{\cong}$ enforces the property $\hat{\mathcal{P}}$ iff $\forall\sigma \in \Sigma^\infty$*

1. *$\hat{\mathcal{P}}(\mathcal{A}(\sigma))$*
2. *$\mathcal{A}(\sigma) \cong \sigma$*

A monitor can correctively$_{\cong}$ enforce a property iff for every possible sequence there exists an equivalent valid sequence which is either finite or has infinitely many valid prefixes, and the transformation into this sequence is computable.

Theorem 1. *A property $\hat{\mathcal{P}}$ is correctively$_{\cong}$ enforceable iff*

1. $\exists \hat{\mathcal{P}}' : (\hat{\mathcal{P}}' \subseteq \hat{\mathcal{P}}) \wedge (\hat{\mathcal{P}}' \subseteq Renewal)$
2. $\hat{\mathcal{P}}$ *is reasonable*
3. *There exists a computable function* $\gamma : \Sigma^{\infty} \rightarrow \hat{\mathcal{P}}' : \forall \sigma \in \Sigma^{\infty} : \gamma(\sigma) \cong \sigma$.
4. $\forall \sigma' \preceq \sigma : \gamma(\sigma') \preceq \gamma(\sigma)$

Proof. ($\Rightarrow$ direction)
By construction of the following automaton. $\mathcal{A} = \langle \Sigma, Q, q_0, \delta \rangle$ where

- $Q = \Sigma^*$, the sequence of actions seen so far.
- $q_0 = \epsilon$
- The transition function δ is given as $\delta(\sigma, a) = (\sigma; a, \sigma')$, where $\sigma = a'; \tau$ and $\gamma(\sigma; a) = \gamma(\sigma); \sigma'$
  Note that from condition 3 of theorem 1 we have that $\gamma(\sigma; a)$ is always defined, and from condition 4 that it will take the recursive form described above.

The automaton maintains the following invariants INV(q): At state $q = \sigma$, $\gamma(\sigma)$ has been output so far, this output is valid and equivalent to σ.
The invariant holds initially, as by definition, ϵ is valid and equivalent to itself. An induction can then show that the invariant is preserved by the transition relation.

($\Leftarrow$ direction) Let $\gamma(\sigma)$ be whatever the automaton outputs on input σ. By definition, γ is a computable function. Furthermore, we have that $\hat{\mathcal{P}}(\sigma)$ and $\gamma(\sigma) \cong \sigma$.

We need to show that the image of γ is a property $\hat{\mathcal{P}}'$ included in $\hat{\mathcal{P}}$ and in renewal. That the image of γ is a subset of $\hat{\mathcal{P}}$ follows trivially from the assumptions $\forall \sigma \in \Sigma^{\infty} : \hat{\mathcal{P}}(\mathcal{A}(\sigma))$. Furthermore, were the output not in renewal, it would include valid sequences with only finitely many valid prefixes. Yet, since the automaton's transition function is restricted to outputing finite valid sequences by the requirement that the finite input be equivalent to the output and equation 4.4 , this is impossible. It follows that the image of γ is a subset of $\hat{\mathcal{P}}$ and renewal. It is also easy to see that $\hat{\mathcal{P}}(\epsilon)$, since if it were not the case, a violation would occur even in the absence of any input action. Finally, since γ is applied recursively to every prefix of the input, it is thus unavoidable that $\forall \sigma' \preceq \sigma : \gamma(\sigma') \preceq \gamma(\sigma)$. □

An equivalence relation $\cong$ over a given set Σ^* can be seen as a set of pairs (x, y), with $x, y \in \Sigma^*$. This allows equivalence relations over the same sets to be compared. Relation $\cong_1$ is a refinement of relation $\cong_2$, noted $\cong_1 < \cong_2$ if the set of pairs in $\cong_1$ is a strict subset of those in $\cong_2$.

Theorem 2. *Let* $\cong_1$, $\cong_2$ *be two equivalence relations and let enforceable*$_{\cong}$ *stand for the set of properties which are correctively*$_{\cong}$ *enforceable, then we have* $\cong_1 < \cong_2$ $\Rightarrow enforceable_{\cong_1} \subset enforceable_{\cong_2}$.

Proof. It is easy to see that any property which is correctively$_{\cong_1}$ enforceable is also correctively$_{\cong_2}$ enforceable, since every pair of sequence that are equivalent w.r.t. $\cong_1$ are also equivalent w.r.t. $\cong_2$. The property can thus be correctively$_{\cong_2}$ enforced using the same transformation function γ as was used in its correctively$_{\cong_1}$ enforcement.

Let $[\sigma]_{\cong}$ stand for the set of sequences equivalent to σ with respect to relation $\cong$. By assumption, there is a σ s.t. $[\sigma]_{\cong_1} \subset [\sigma]_{\cong_2}$. Let $\hat{\mathcal{P}}$ be the property defined s.t. $\neg\hat{\mathcal{P}}(\tau) \Leftrightarrow \tau \in [\sigma]_{\cong_1}$. This property is not correctively$_{\cong_1}$ enforceable as there exists no valid equivalent sequences which the monitor can output when its input is σ. The property can be correctively$_{\cong_2}$ enforced by outputting a sequence in $[\sigma]_{\cong_2} \backslash [\sigma]_{\cong_1}$ when the input is σ. □

It follows from this theorem that the coarser the equivalence relation used by the monitor is, the greater the set of enforceable$_{\cong}$ properties.

The following lemma is used in setting an upper bound to the set of enforceable properties.

Lemma 3. *Let* $\cong$ *be an equivalence relation and* $\hat{\mathcal{P}}$ *be some correctively*$_{\cong}$ *enforceable property. Then, for all* $\hat{\mathcal{P}}$*' s.t.* $\hat{\mathcal{P}} \subseteq \hat{\mathcal{P}}'$ *we have that* $\hat{\mathcal{P}}$*' is correctively*$_{\cong}$ *enforceable.*

The monitor has only to simulate it's enforcement of $\hat{\mathcal{P}}$ in order to correctively$_{\cong}$ enforce $\hat{\mathcal{P}}$'.

6 Equivalence Relations

In this section, we consider two examples of the equivalence relation $\cong$, and examine the set of properties enforceable by each.

6.1 Factor Equivalence

The first equivalence relation we will consider is factor equivalence, which models the class of transactional properties introduced in section 3. A word $\tau \in \Sigma^*$ is a factor of a word $\omega \in \Sigma^\infty$ if $\omega = \upsilon; \tau; \upsilon'$, with $\upsilon \in \Sigma^*$ and $\upsilon' \in \Sigma^\infty$. Two sequences τ, τ' are factor equivalent, w.r.t. a given set of valid factors $\mathcal{T} \subseteq \Sigma^*$ if they both contain the same multiset of factors from $\mathcal{T}$. We use a multiset rather than simply comparing the set of factors from $\mathcal{T}$ occurring in each sequence so as to be able to distinguish between sequences containing a different number of occurrences of the same subset of factors. This captures the intuition that if certain valid transactions are present in the input sequence, they must still be present in the output sequence, regardless of any other transformation made to ensure compliance with the security property. In this context, the desired behavior of the system can be defined by a multiset of valid transactions. A valid run of this system consists of a finite or infinite sequence of well-formed transactions, while an invalid sequence is a sequence containing malformed or incomplete transactions. One may reasonably consider all sequences exhibiting the same multiset of valid transactions to be equivalent to each other. Transactional properties form a subset to the class of renewal properties which can be effectively$_=$ enforced [13], which allows the longest valid prefix to be output [14]. In [5], Bielova et al. propose an alternate enforcement paradigm, which allows all valid transactions to be output. Corrective$_\sqsubseteq$ enforcement can be seen as a generalization of their work.

Let $valid_\mathcal{T}(\sigma)$, which stand for the multiset of factors of from the sequence σ which are present in $\mathcal{T}$, be the abstraction function $\mathcal{F}$. The partial order $\sqsubseteq$ used to correctively enforce this property is thus given as $\forall \sigma, \sigma' \in \Sigma^\infty : \sigma \sqsubseteq \sigma' \Leftrightarrow valid_\mathcal{T}(\sigma) \subseteq valid_\mathcal{T}(\sigma')$. This partial order captures the intuition that any valid transaction present in the original sequence must also be present in the monitor's output.

For example, let $\Sigma = \{\texttt{open}, \texttt{close}, \texttt{log}\}$ be a set of atomic actions and let $\mathcal{T} = \{\texttt{open}; \texttt{log}; \texttt{close}\}$ be the set containing the only allowed transaction. If the input sequence is given as $\sigma = \texttt{log}; \texttt{open}; \texttt{log}; \texttt{close}; \texttt{log}; \texttt{open}; \texttt{close}; \texttt{open}; \texttt{log}; \texttt{close}$, then $valid_\mathcal{T}(\sigma)$ is the multiset containing two instances of the factor $\texttt{open}; \texttt{log}; \texttt{close}$.

Intuitively, a sequence is smaller than another on the partial order if it has strictly fewer transactions, and two sequences are equivalent if they share the same valid transactions.

We now turn our attention to the set of properties that are correctively$_{\cong_\mathcal{T}}$ enforceable. Intuitively, a monitor can enforce this property by first suppressing the execution until it has seen a factor in $\mathcal{T}$, at which point the factor is output, while any invalid transaction is suppressed. This method of enforcement is analogous to the one described in [5] as delayed all-or-nothing enforcement. Any sequence output in this manner would preserve all its factors in $\mathcal{T}$, and thus be equivalent to the input sequence, but is composed of a concatenation of factors from $\mathcal{T}$, and hence is valid.

Let $\mathcal{T} \subseteq \Sigma^*$ be a set of factors and let $\hat{\mathcal{P}}_\mathcal{T}$ a transactional property as defined in section 3. Note first that all properties enforceable by this approach are in renewal, as they are formed by a concatenation of valid finite sequences. Also, the property necessarily must be reasonable, (i.e. $\hat{\mathcal{P}}(\epsilon)$) as the monitor will not output anything if the input sequence does not contain any factors in $\mathcal{T}$. Finally, for the property $\hat{\mathcal{P}}_\mathcal{T}$ to be correctively$_{\cong_\mathcal{T}}$ enforceable in the manner described above, the following restriction, termed unambiguity must be imposed on $\mathcal{T}$:

$$\forall \sigma, \sigma' \in \mathcal{T} : \forall \tau \in pref(\sigma) : \forall \tau' \in suf(\sigma') : \tau \neq \epsilon \wedge \tau' \neq \epsilon \Rightarrow \tau; \tau' \notin \mathcal{T} \qquad \text{(unambiguity)}$$

To understand why this restriction is necessary, consider what would happen in its absence: it would be possible for the monitor to receive as input a sequence which can be parsed either as the concatenation of some valid transactions, or as a different valid transaction bracketed with invalid factors. That is, let $\sigma_1;\sigma_2 = \tau_1;\sigma_3;\tau_2$ be the monitor's input, with $\sigma_1, \sigma_2, \sigma_3 \in \mathcal{T}$ and $\tau_1, \tau_2 \notin \mathcal{T}$. If the monitor interprets the sequence as a concatenation of the valid transactions σ_1 and σ_2, then it has to preserve both factors in its output. However, if it parses the sequence as $\tau_1;\sigma_3;\tau_2$, then it must output only the equivalence sequence σ_3. Since the two sequences are syntactically identical, the monitor has no information of which to base such a decision.

Theorem 4. *A property $\hat{\mathcal{P}}_{\mathcal{T}}$ correctively$_{\cong_{\mathcal{T}}}$ enforceable if it is transactional, reasonable, and $\mathcal{T}$ is unambiguous.*

Proof. The proof has been omitted out of space considerations. It is available from the authors upon request. □

We have only to refer to lemma 3 in order to state a precise upper bound to the set of enforceable properties.

Theorem 5. *A property $\hat{\mathcal{P}}$ is correctively$_{\cong_{\mathcal{T}}}$ enforceable iff $\hat{\mathcal{P}}_{\mathcal{T}} \subseteq \hat{\mathcal{P}}$ and $\mathcal{T}$ is unambiguous.*

Proof. The proof has been omitted out of space considerations. It is available from the authors upon request. □

6.2 Prefix Equivalence

In this section, we show that Ligatti et al.'s result from [13], namely that the set of properties effectively$_=$ enforceable by an edit automaton corresponds to the set of reasonable renewal properties with a computability restriction added[2], can be stated as a special case of our framework.

First, we need to align our definitions of enforcement. Using effective enforcement, they only require that the monitor's output be equivalent to its input when the latter is valid, and while placing no such restriction on the output otherwise. The semantics of their monitor however, do impose that the output remain a prefix of the input in all cases, and indeed, that the longest valid prefix always be output (see [8]). This characterization can be translated in our formalism by instantiating $\cong$ to $\cong_{\preceq} \overset{def}{=} \forall \sigma, \sigma' \in \Sigma^* : \sigma \cong_{\preceq} \sigma' \Leftrightarrow pref(\sigma) \cap \hat{\mathcal{P}} = pref(\sigma') \cap \hat{\mathcal{P}}$. Using this relation, two sequences are equivalent, w.r.t. a given property $\hat{\mathcal{P}}$ iff they have the same set of valid prefixes.

Theorem 6. *A property $\hat{\mathcal{P}}$ is effectively$_=$ enforceable iff it is correctively$_{\cong_{\preceq}}$ enforceable.*

[2] Actually, the authors identified a corner case in which a property not in the set described above. This occurs when the monitor reaches a point where only one valid continuation is possible. The input can then be ignored and this single continuation is output. We have neglected to discuss this case here as it adds comparatively little to the range of enforceable properties.

Proof. The proof has been omitted out of space considerations. It is available from the authors upon request. □

It would be intuitive to instantiate the partial order $\sqsubseteq$ to $\preceq$. Other possibilities can be considered, which would more closely follow the specific property being enforced.

Theorem 7. *A property $\hat{\mathcal{P}}$ is correctively$_{\cong_{\preceq}}$ enforceable iff it is in renewal, reasonable and computable.*

Proof. Immediate from theorem 6 and theorem 3 of [13]. □

As discussed in [13], this set includes a wide range of properties, including all safety properties, some liveness properties such as the "eventually audits" properties requiring that an action eventually be logged, and properties which are neither safety nor liveness such as the transactional properties described in section 4. Furthermore, if the behavior of the target system is known to consist only of finite executions, then every sequence is in renewal.

7 Nonuniform Enforcement

In this section, we investigate the possibility of extending the set of enforceable properties by giving the monitor some knowledge of the target program's possible behavior. This question was first raised in [16]. In [4], the authors distinguish between the uniform context, in which the monitor must consider that every sequence in Σ^∞ can occur during the target program's execution, from the nonuniform context, in which the set of possible executions is a subset of Σ^∞. They further show that in some case, the set of properties enforceable in a nonuniform context is greater than that which is enforceable in a uniform context. Later Chabot et al. [7] showed that while this result did not apply to all runtime enforcement paradigms, it did apply to that of truncation-based monitor. Indeed, they show that in this monitoring context, a monitor operating with a subset of Σ^∞ is always more powerful than one which considers that every sequence can be output by its target.

Let $\mathcal{S}$ stand for the set of sequences which the monitor considers as possible executions of the target program. $\mathcal{S}$ is necessarily an over approximation, built from static analysis of the target. We write correctively$^{\mathcal{S}}_{\cong}$ enforceable, or just enforceable$^{\mathcal{S}}_{\cong}$, to denote the set of properties that are correctively$_{\cong}$ enforceable, when only sequences from $\mathcal{S} \subseteq \Sigma^\infty$ are possible executions of the target program. A property is correctively$^{\mathcal{S}}_{\cong}$ enforceable iff for every sequence in $\mathcal{S}$, the monitor can return a valid and equivalent sequence.

Definition 4. *Let $\mathcal{A}$ be an edit automaton and let $\mathcal{S} \subseteq \Sigma^\infty$ be a subset of executions. $\mathcal{A}$ correctively$^{\mathcal{S}}_{\cong}$ enforces the property $\hat{\mathcal{P}}$ iff $\forall \sigma \in \mathcal{S}$*

1. $\hat{\mathcal{P}}(\mathcal{A}(\sigma))$
2. $\mathcal{A}(\sigma) \cong \sigma$

Theorem 8. *A property $\hat{\mathcal{P}}$ is correctively$^{\mathcal{S}}_{\cong}$ enforceable iff*

1. *$\hat{\mathcal{P}}$ is Reasonable*
2. *$\exists \hat{\mathcal{P}}' \subseteq \hat{\mathcal{P}} : \hat{\mathcal{P}}' \in renewal : (\exists \gamma \in \mathcal{S} \rightarrow \hat{\mathcal{P}}' : (\forall \sigma \in \mathcal{S} : \gamma(\sigma) \cong \sigma) \wedge (\forall \sigma, \sigma' \in \mathcal{S} : \sigma' \preceq \sigma \Rightarrow \gamma(\sigma') \preceq \gamma(\sigma)) \wedge \gamma$ is computable$)$*

Proof. The proof follows exactly as that of Theorem 1. □

Lemma 9. *Let $\mathcal{S} \subseteq \Sigma^{\infty}$ and $\hat{\mathcal{P}}$ be a reasonable property $\hat{\mathcal{P}}$ is trivially correctively$^{\mathcal{S}}_{\cong}$ enforceable iff $\mathcal{S} \subseteq \hat{\mathcal{P}}$. If this is the case, the monitor can enforce the property by always returning the input sequence.*

We assume that $\mathcal{S}$ represent an upper approximation of a program executions set, determined by static analysis. It would be desirable if the set of enforceable properties increased monotonously each time a sequence was removed from $\mathcal{S}$. This means that any effort made to perform or refine a static analysis of the target program would payoff in the form of an increase in the set of enforceable properties. This is unfortunately not the case. As a counterexample, consider the equivalence relation defined as $\forall \sigma, \sigma' \in \Sigma^{\infty} : \sigma \cong \sigma'$. It is obvious that any satisfiable property can be trivially enforced in this context, simply by always outputting any valid sequence, which is necessarily equivalent to the input. No benefit can then be accrued by restricting $\mathcal{S}$.

There are, of course, some instances where constraining the set $\mathcal{S}$ does result in an increase in the set of correctively$^{\mathcal{S}}_{\cong}$ enforceable properties. This occurs when invalid sequences with no valid equivalent are removed from $\mathcal{S}$. Indeed, for any subsets, $\mathcal{S}, \mathcal{S}'$ of Σ^{∞} s.t. $\mathcal{S} \subset \mathcal{S}' \wedge \mathcal{S}' \backslash \mathcal{S} \neq \{\epsilon\}$, there exists an equivalence relation $\cong$ for which enforceable$^{\mathcal{S}'}_{\cong} \subset$ enforceable $\subset^{\mathcal{S}}_{\cong}$.

Theorem 10. *Let $\mathcal{S} \subset \mathcal{S}' \subseteq \Sigma^{\infty} \wedge \mathcal{S}' \backslash \mathcal{S} \neq \{\epsilon\}$. There exists an equivalence relation $\cong$ s.t. enforceable$^{\mathcal{S}'}_{\cong} \subset$ enforceable $\subset^{\mathcal{S}}_{\cong}$.*

Proof. Let $\cong$ be defined s.t. $\exists \sigma \in \mathcal{S}' \backslash \mathcal{S} : [\sigma] \cap \mathcal{S} \neq \emptyset$. Let $\hat{\mathcal{P}}$ be the property defined as $\hat{\mathcal{P}}(\sigma) \Leftrightarrow (\sigma \notin \mathcal{S} \wedge \sigma \neq \epsilon)$. This property is not enforceable$^{\mathcal{S}'}_{\cong}$ since there exists sequences in $\mathcal{S}'$ with no valid equivalent. The property is trivially enforceable $\subset^{\mathcal{S}}_{\cong}$. □

A final question of relevance on the topic of nonuniform enforcement is whether there exists some equivalence relations $\cong$ for which every reduction of the size of $\mathcal{S}$ monotonously increases the set of properties that are correctively$^{\mathcal{S}}_{\cong}$ enforceable. In other words, if there exists some $\cong$ for which $\mathcal{S} \subset \mathcal{S}' \Rightarrow$ enforceable$^{\mathcal{S}'}_{\cong} \subset$ enforceable$^{\mathcal{S}}_{\cong}$. Anyone operating under such an equivalence relation would have an added incentive to invest in static analysis of the target, as he or she would be guaranteed an increase in the set of enforceable properties. Unfortunately, it can be shown that this result holds only when $\cong$ is syntactic equality and at least one sequence different from ϵ is removed from the set of possible sequences.

Theorem 11. $(\sigma \cong \sigma \Leftrightarrow \sigma = \sigma') \Leftrightarrow \forall \mathcal{S}, \mathcal{S}' \subseteq \Sigma^{\infty} : (\mathcal{S} \subset \mathcal{S}' \wedge \mathcal{S}' \backslash \mathcal{S} \neq \{\epsilon\} \Rightarrow$ *enforceable*$^{\mathcal{S}'}_{\cong} \subset$ *enforceable*$^{\mathcal{S}}_{\cong})$

Proof. ($\Rightarrow$ direction) Let $\hat{\mathcal{P}}$ be defined such that $\hat{\mathcal{P}}(\sigma) \Leftrightarrow (\sigma \notin \mathcal{S}' \backslash \mathcal{S})$. This property cannot be correctively$^{\mathcal{S}'}_{\cong}$ enforceable since any sequence in $\mathcal{S}' \backslash \mathcal{S}$ does not have a valid equivalent. The property is trivially correctively$^{\mathcal{S}}_{\cong}$ enforceable.
($\Leftarrow$ direction) By contradiction, let $\cong$ be different than syntactic equality. This implies there exists $\sigma, \sigma' \in \mathcal{S}' : \sigma \cong \sigma' \wedge \sigma \neq \sigma'$. Further, let $\mathcal{S}' = \{\sigma, \sigma'\}$ and $S = \{\sigma\}$. We show that any property that is correctively$^{\mathcal{S}}_{\cong}$ enforceable is also correctively$^{\mathcal{S}'}_{\cong}$ enforceable. There are five cases to consider. :

- $\sigma, \sigma' \in \hat{\mathcal{P}}$: In this case, the property is always trivially enforceable.
- $\sigma \in \hat{\mathcal{P}} \wedge \sigma' \notin \hat{\mathcal{P}}$: Such a property would be both correctively$^{\mathcal{S}}_{\cong}$ enforceable and correctively$^{\mathcal{S}'}_{\cong}$ enforceable by automaton $\mathcal{A}$ for which $\mathcal{A}(\tau) = \sigma$ for all τ in the input set.
- $\sigma' \in \hat{\mathcal{P}} \wedge \sigma \notin \hat{\mathcal{P}}$: Such a property would be both correctively$^{\mathcal{S}}_{\cong}$ enforceable and correctively$^{\mathcal{S}'}_{\cong}$ enforceable by automaton $\mathcal{A}$ for which $\mathcal{A}(\tau) = \sigma'$ for all τ in the input set.
- $\sigma, \sigma' \notin \hat{\mathcal{P}} \wedge \exists \sigma'' \cong \sigma : \hat{\mathcal{P}}(\sigma'')$: Such a property would be both correctively$^{\mathcal{S}}_{\cong}$ enforceable and correctively$^{\mathcal{S}'}_{\cong}$ enforceable by automaton $\mathcal{A}$ for which $\mathcal{A}(\tau) = \sigma''$ for all τ in the input set.
- $\sigma, \sigma' \notin \hat{\mathcal{P}} \wedge \neg \exists \tau \cong \sigma : \hat{\mathcal{P}}(\tau)$: This property can neither be correctively$^{\mathcal{S}}_{\cong}$ enforceable nor can it be correctively$^{\mathcal{S}'}_{\cong}$ enforceable since there exists some sequences with no valid equivalent.

Finally, observe that since only reasonable sequences are enforceable, no possible gain can be accrued from removing only ϵ from the set of possible sequences. □

8 Conclusion and Future Work

In this paper, we propose a framework to analyze the security properties enforceable by monitors capable of transforming their input. By imposing constraints on the enforcement mechanism to the effect that some behaviors existing in the input sequence must still be present in the output, we are able to model the desired behavior of real-life monitors in a more realistic and effective way. We also show that real life properties are enforceable in this paradigm, and give prefix equivalence and factor equivalence as possible examples of realistic equivalence relations which could be used in a monitoring context. The set of properties enforceable using these two equivalence relations is related to previous results in the field.

Future work will focus on other equivalence relations. Two meaningful equivalence relations which we are currently studying are subword equivalence and permutation equivalence. The first adequately models the behavior of a monitor that is allowed to insert actions into the program's execution, but may not subtract anything from it. The second models the behavior of a monitor which can reorder the actions performed by its target, but may not add or remove any of them. An even more general framework that could be envisioned would be one in which the behavior that the monitor must preserve is stated in a temporal logic.

References

1. Alpern, B., Schneider, F.B.: Recognizing safety and liveness. Distributed Computing 2, 17–126 (1987)
2. Alpern, B., Schneider, F.: Defining liveness. Information Processing Letters 21(4), 181–185 (1985)
3. Bauer, A., Leucker, M., Schallhart, C.: Monitoring of real-time properties. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 260–272. Springer, Heidelberg (2006)
4. Bauer, L., Ligatti, J., Walker, D.: More enforceable security policies. In: Proceedings of the Foundations of Computer Security Workshop (July 2002)
5. Bielova, N., Massacci, F., Micheletti, A.: Towards practical enforcement theories. In: Knapskog, S.J. (ed.) NordSec 2009. LNCS, vol. 5838, pp. 239–254. Springer, Heidelberg (2009)
6. Chabot, H.: Sécurisation de code basée sur la combinaison d'analyse statique et dynamique, génération de moniteur partir d'un automate de Rabin. Master's thesis, Laval University (2008)
7. Chabot, H., Khoury, R., Tawbi, N.: Generating in-line monitors for Rabin automata. In: Jøsang, A., Maseng, T., Knapskog, S.J. (eds.) NordSec 2009. LNCS, vol. 5838, pp. 287–301. Springer, Heidelberg (2009)
8. Falcone, Y., Fernandez, J.C., Mounier, L.: Enforcement monitoring wrt. the safety-progress classification of properties. In: Proceedings of the ACM Symposium on Applied Computing (SAC), pp. 593–600. ACM Press, New York (March 2009)
9. Fong, P.: Access control by tracking shallow execution history. In: Proceedings of the 2004 IEEE Symposium on Security and Privacy (May 2004)
10. Hamlen, K.W., Morrisett, G., Schneider, F.B.: Computability classes for enforcement mechanisms. ACM Transactions on Programming Languages and Systems (TOPLAS) 28(1), 175–205 (2006)
11. Lamport, L.: Proving the correctness of multiprocess programs. IEEE Transactions on Software Engineering 3(2), 125–143 (1977)
12. Ligatti, J., Bauer, L., Walker, D.: Edit automata: Enforcement mechanisms for run-time security policie. International Journal of Information Security (2004)
13. Ligatti, J., Bauer, L., Walker, D.: Enforcing non-safety security policies with program monitors. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 355–373. Springer, Heidelberg (2005)
14. Ligatti, J., Bauer, L., Walker, D.: Run-time enforcement of nonsafety policies. ACM Transactions on Information and System Security 12(3), 1–41 (2009)
15. Ligatti, J., Reddy, S.: A theory of runtime enforcement, with results. Tech. Rep. USF-CSE-SS-102809, University of South Florida (April 2010)
16. Schneider, F.B.: Enforceable security policies. Information and System Security 3(1), 30–50 (2000)
17. Syropoulos, A.: Mathematics of multisets. In: Calude, C.S., Pun, G., Rozenberg, G., Salomaa, A. (eds.) Multiset Processing. LNCS, vol. 2235, pp. 347–358. Springer, Heidelberg (2001)
18. Talhi, C., Tawbi, N., Debbabi, M.: Execution monitoring enforcement under memory-limitations constraints. Information and Computation 206(1), 158–184 (2008)

Model Checking of Location and Mobility Related Security Policy Specifications in Ambient Calculus

Devrim Unal[1], Ozan Akar[2], and M. Ufuk Caglayan[2]

[1] TUBITAK National Research Inst. of Electronics and Cryptology
[2] Bogazici University

Abstract. Verification of security for mobile networks requires specification and verification of security policies in multiple-domain environments. Mobile users present challenges for specification and verification of security policies in such environments. Formal methods are expected to ensure that the construction of a system adheres to its specification. Formal methods for specification and verification of security policies ensure that the security policy is consistent and satisfied by the network elements in a given network configuration. We present a method and a model checking tool for formal specification and verification of location and mobility related security policies for mobile networks. The formal languages used for specification are Predicate Logic and Ambient Calculus. The presented tool is capable of spatial model checking of Ambient Calculus specifications for security policy rules and uses the NuSMV model checker for temporal model checking.

Keywords: model checking, ambient calculus, security policy.

1 Introduction

Resource sharing and provision of services in networks with multiple administrative domains is an ever-increasing need. Roaming is another concept that comes into consideration when dealing with multi-domain resource sharing applications where users are allowed to use network connectivity of multiple domains. Roaming means that users are able to connect to and use networks of multiple administrative domains. Security management in such an environment requires specification of inter-domain security policies and cross-domain administration of security mechanisms. Authorization mechanisms determine the access rights for a user based on the security policy. The access control mechanisms then control the user access to the resource based upon these determined access rights. The user actions should be verified against home and visited domain policies as they access resources on visited domains. Formal verification can be used to ensure that visiting users are not bypassing security mechanisms and violating security policy by making use of the internal trust relationships.

In this paper, we propose a method and a model checking tool for formal specification and verification of multi-domain security policies with location and

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 155–168, 2010.

mobility constraints. Appropriate to the nature of multi-domain mobile networks, the proposed method focuses on the location and mobility aspects of security policies. The formalism of the method is based on predicate logic, ambient calculus and its ambient logic. An ambient calculus model checker capable of spatial and temporal model checking has been built for the implementation of proposed method. The model checker presents novel computational methods for decreasing the time and space complexity of spatial model checking. The model checking approach complements our previous work on use of theorem proving for security policies [19].

2 Formal Languages and Methods for Specification and Verification of Policies

Logic-based security policy models provide a general framework for security policy specification. Becker et al.'s SECPAL [1] is a formal security policy language for Grid environments. The Flexible Authorization Framework (FAF) is a logic programming based method for definition, derivation and conflict resolution of authorization policies [14,13]. Another study based on logic that supports explicit denials, hierarchies, policy derivation and conflict resolution is [2]. Ponder [9] is a general purpose formal security policy language. Woo and Lam [20] define a paraconsistent formal language for authorizations based on logical constructs. In [8] deontic logic is used for modeling the concepts of permission, obligation and prohibition with organizational constructs. A security policy language based on the set-and-function formalism is presented in [16].

Model checking and theorem proving have been applied for verification of security policies. Acpeg [21] is a tool for evaluating and generating access control policies based on first-order logic. We have previously applied theorem proving to verification of security policies. In [19] we use Coq for checking that an authorisation security policy is conflict-free, initially and as authorisation rules are added and removed, while [10] uses first order linear temporal logic embedded within Isabelle to formalise and verify RBAC authorisations constraints. A more recent study, [18] uses nontemporal and history-based authorization constraints in the Object Constraint Language (OCL) and first-order linear temporal logic (LTL) to verify role-based access control policies with the help of a theorem prover.

Ambient Calculus [4] has been used for modeling and reasoning about security in mobile systems. Reasoning about spatial configurations for application level security policies in ubiquitous environments is one of the issues investigated in Scott's PhD thesis [17]. In this study a simplified version of ambient calculus and ambient logic is used in policy rules of a security policy. BACI$_R$ [7] is a boxed ambient calculus with RBAC mechanisms used to define access control policies for ambients. Similarly, in our approach, the Ambient Calculus and Ambient Logics [3] are utilized. In contrast to BACI$_R$ which places policies inside Ambient Calculus formulas, we use Ambient Calculus for specification of processes and Ambient Logics for specification of policies and complement them with Predicate

Logic based relational model. In contrast to Scott's approach, network level policies rather than application level policies will be covered and locations will denote placement in domains and hosts. For model checking of ambient calculus specifications, our approach is similar to the work of Mardare et al [15]. We use a modified version of Mardare algorithm and present an algorithm based on use of capability trees that reduces complexity of state space generation and matching of ambient logic formulas to states. In the works of Charatonik et al. [5,6] exhaustive search is offered for searching possible decompositions of processes and searching sub locations when checking spatial modalities. We offer heuristics for searching possible decompositions of processes and searching sub locations to reduce the search space.

3 A Formal Model for Security Policies for Multi-domain Mobile Networks in Ambient Calculus

3.1 Formal Model for Security Policy

Access Control Model: The access control model is specified using Predicate Calculus and First Order Set Theory. The access control model is based on the RBAC [11] model. The Hierarchical RBAC model extends the Core RBAC model with role hierarchies. We use the hierarchical RBAC model that supports role hierarchies. For introduction of location and mobility constraints into the security policy, we extend the Hierarchical RBAC model by adding Domains, Hosts, Object Types, Conditions and Location Constraints. The concept of sessions are not utilized in our model.

- Constants:
  d, n, m, o, t, v: Number of domains, hosts, users, roles, objects and object types, respectively.
- Sets:
  - $D = \{D_1, D_2, ...D_d\}$: Domains , $H = \{H_1, H_2, ...H_n\}$: Hosts
  - $U = \{U_1, U_2, ...U_m\}$: Users , $R = \{R_1, R_2, ...R_o\}$: Roles
  - $O = \{O_1, O_2, ..., O_t\}$: Objects , $OT = \{OT_1, OT_2, ...OT_v\}$: Object Types
- Relations:
  $HOD : H \times D$: Maps hosts to domains. $HOD(H_i, D_a)$ denotes that H_i is enrolled to Domain D_a.
  $UOD : U \times D$: Maps users to domains. $UOD(U_j, D_a)$ denotes that U_j is enrolled to Domain D_a.
  $OOT : O \rightarrow OT$:Function that specifies the type of an object. $OOT(O_k)$ gives the type of object O_k.
  $UA : U \times R$: Relation for assignment of users to roles.
  $PA : R \times AO \times SA$: Relation for associating roles with permissions.

Authorization Terms and Security Policy
An Authorization Term is of the form $at = (as, ao, sa, fo, co)$ where $as \in$AS, ao $\in$ AO, $sa \in S \times A$,fo: a formula, where formula is an ambient logic formula, co: a condition, where condition is a predicate logic formula. Security policy is a set of authorization terms.

- Sets:
  AS: $AS = U \cup R$. The set of Authorization Subjects. Authorization Subjects are active entities that may conduct an Action on an Authorization Object.
  AO: $AO = O \cup OT \cup H \cup D$. The set of Authorization Objects. The authorization object is the entity upon which an action is conducted.
  A : Set of actions conductable by subjects on objects. We take A to be fixed in this study:
  A ={*Enroll, Login, Logout, Execute, Read, Write, Send, Receive, Delete, Create*}
  Signs: $S = \{+, -\}$ Represents permission or denial.
  Signed Actions: $S \times A$: Represents permission or denial of an action (+,read) denotes that read action is permitted.
- Predicates:
  - EnrolledDomainHost (*host, domain*): *host* is a registered member of the Domain *domain*
  - EnrolledDomainUser (*user, domain*): *user* is a registered member of *domain*
  - ActiveDomainUser (*user, domain*): *user* has logged into a *domain*
  - RoleAllowed (*user, role*): *user* has assumed the Role of *role*
  - ActionAllowed (*as, action*): Authorization Subject *as* is allowed to execute action *action*
- Conditions: First-order sentences built on the Predicates defined above.
- Spatial Formula: Ambient Logic formula that includes names of domains, authorization subjects and authorization objects. The formula will be described in the following sections.

3.2 Formal Specification of Mobile Processes

Ambient calculus, proposed by Cardelli and Gordon, is a process calculus which is able to theorize about concurrent systems that include mobility and locations [4]. The proposed methodology uses ambient calculus for specifying multi-domain mobile network configurations. Fragment of ambient calculus used in this paper is shown at Table 1. The semantics of ambient calculus is based on structural congruence relation.

The formal model for mobility is a finite fragment of the ambient calculus with public names as used in [6]. In the formal specification, domains, hosts, users and objects are modeled as Ambients. The actions are modeled as Ambient Calculus capabilities. A process specification shows a trace of a process in a certain mobile network scenario. Each scenario may be modeled as a set of process specifications. These specifications will then be checked against a security policy for compliance. The process specification involves capabilities, objects and ambients. Resources may be *input* and *output* by the ambients. The ambients may be World, Domains, Hosts and Users. Below some examples of object, host and user mobility specification of mobility as ambient calculus processes are listed. Some known notation conventions are utilized: for example $n[]$ means $n[0]$. The symbol → represents the reduction relation and →* represents a series of reductions.

Table 1. Mobility and communication primitives of Ambient Calculus

P, Q	$::=$	processes	M	$::=$	capabilities
	0	inactivity		x	variable
	$P \mid Q$	composition		n	name
	$M[P]$	ambient		$in\ \ M$	can enter M
	$M.P$	capability		$out\ \ M$	can exit M
	$(x).P$	input		$open\ \ M$	can open M
	$\langle M \rangle$	asynchronous output		ϵ	null
				$M.M$	path

- *File1* is copied to *Portable1*:

$$\mathit{World}[\mathit{DomainA}[\mathit{Server1}[\text{folder }[\text{out folder. out } \mathit{Server1}. \text{ in } \mathit{Portable1}. \text{ in folder. } \mathit{File1}]] \mid \mathit{File1}\ []]] \mid \mathit{Portable1}[\text{folder}[]]]] \rightarrow^{*}$$
$$\mathit{World}[\mathit{DomainA}[\mathit{Server1}[\text{folder }[\mathit{File1}]]]] \mid \mathit{Portable1}[\text{folder}[\mathit{File1}[]]]]]$$

- A message M is sent from *User1* to *User3*:

$$\mathit{World}\ [\mathit{DomainA}\ [\mathit{Server1}\ [\mathit{User1}\ [\text{message}[M \mid \text{out } \mathit{User1}. \text{ out } \mathit{Server1}. \text{ out } \mathit{DomainA}. \text{ in } \mathit{DomainB}, \text{ in } \mathit{Client2}. \text{ in } \mathit{User3}.0]]]] \mid \mathit{DomainB}\ [\mathit{Client2}\ [\mathit{User3}\ [\text{open message}.(m).0]]]] \rightarrow^{*}$$
$$\mathit{World}\ [\mathit{DomainA}[\mathit{Server1}[\mathit{User1}[]]] \mid \mathit{DomainB}[\mathit{Client2}[\mathit{User3}[M]]]]$$

We also provide the mapping of actions in the security policy model to Ambient Calculus specifications. These are provided as a template and based on inference of specific subject and object names from the high-level specifications of security policy, the model checking tool is presented with suitable Ambient Calculus specifications.

$$\mathit{Enroll} =_{\mathrm{def}} \mathit{newz.indomain.z}[] \mid \mathit{domain}[], \mathit{wherez} \in U \cup H, \mathit{domain} \in D \tag{1}$$

$$\mathit{Login} =_{\mathrm{def}} \mathit{domain}[z[\mathit{inhost}] \mid \mathit{host}[]], \mathit{wherez} \in U, \mathit{host} \in H, \mathit{domain} \in D \tag{2}$$

$$\mathit{Logout} =_{\mathrm{def}} \mathit{domain}[\mathit{host}[z[\mathit{outhost}]]], \mathit{wherez} \in U, \mathit{host} \in H, \mathit{domain} \in D \tag{3}$$

3.3 Formalization of Location and Mobility Related Actions in Authorization Term

The spatial formula in the Authorization Term is specified using the Ambient Logic[3]. Fragment of ambient logic used in this paper is shown in Table 2. Ambient logic has temporal and spatial modalities in addition to propositional logic elements. Semantics of the connectives of the ambient logic are given through satisfaction relations defined in [3]. The definition of satisfaction is based heavily on the structural congruence relation. The satisfaction relation is denoted by $\models$ symbol. To express that process P satisfies the formula $\mathscr{A}$, $P \models \mathscr{A}$ is used. The

Table 2. Syntax of ambient logic

η		a name n		
$\mathscr{A},\mathscr{B},\mathscr{C}$ ::=	T	true	$\mathscr{A}\vert\mathscr{B}$	composition
	$\neg\mathscr{A}$	negation	$n[\mathscr{A}]$	location
	$\mathscr{A}\vee\mathscr{B}$	disjunction	$\Diamond\mathscr{A}$	sometime modality
	0	void	$\diamond\mathscr{A}$	somewhere modality

symbol Π denotes the set of processes, Φ denotes the set of formulas, ϑ denotes the set of variables, and Λ denotes the set of names.

The possibility of conflicts arising of conflicting actions are resolved using the theorem prover as presented in our previous work in [19] before presenting rules to the model checker. Using the formalization methodology for authorization terms and spatial formula described above, some example security policy definitions with location constraints, which can be specified with our formal authorization terms are presented below.

1. All allowed users can read files in folder *Project_ Folder*, if they are in a location that contains this folder: (as = *, ao = *Project_ Folder*, sa = + *read*, co = *ActionAllowed* (*as*, *sa*), fo = ◇(*as*[] | *ao*[]))
2. All allowed users can send *E-mail* between the *UniversityA* and *UniversityB* domains: (as = *, ao = *E-mail*, sa = + *send*, co = *ActionAllowed* (*as*, *sa*), fo = *UniversityA* [◇*as*[]] | *UniversityB* [◇*ao*[]] \/ *UniversityB* [◇*as*[]] | *UniversityA*[◇*ao*[]])

To check location constraints in security policy, the input to the model checker tool is an Ambient Calculus specification and a set of Ambient Logic formulas. An example scenario specified in Ambient Calculus and a security policy rule specified in Ambient Logic is presented below. In this example there are two domains, *Domain1* and *Domain2*, where *User2* is mobile and tries to read data from *File1* by logging into *Host1*. Spatial formula in the policy rule states that *Host2* can not contain *Data1* and *Data2* at the same time. This is a rule that means *Domain2* data should not be copied to *Domain1*.

- Ambient Calculus Specification: *Domain1* [*User1*[] | *Host1* [*File1* [*Data1* | in *User2*.0 | out *User2*.0]]]] | *Domain2* [*Host2* [*User2* [out *Host2*.0 | out *Domain2*.0 | in *Domain1*.in *Host1*.0 | out *Host1*.out *Domain1*.0 | in *Domain2*.in *Host2*.0 in *File1*.0 | in *File2*.0 | out *File1*.0 | out *File2*.0] | *File2*[*Data2*[]]]]
- Ambient Logic Specification: □ { ¬ ◇ { ◇ *Host2*[◇ {*Data1*[T] | *Data2*[T]}] | T}

4 Model Checking of Security Policy Specifications in Ambient Calculus Model Checker

The general structure of the Ambient Calculus model checker is given in Figure 1. To benefit from existing methodologies we divide our problem into two sub problems as temporal model checking and spatial model checking. The temporal model checker is used for carrying out satisfaction process for the Sometime and Everytime connectives of ambient logic. The proposed model checking method generates all possible future states and build a state transition system based on the Ambient Calculus process specification. After evaluation of Ambient Logic formula in each state, this state transition system is processed into a Kripke Structure (Definition 4) which is then given to temporal model checker. NuSMV [12] is used as a temporal model checker. Outline of the proposed algorithm for the model checking problem is below.

1. Define atomic propositions with respect to spatial properties of ambient logic formula and register the (atomic proposition-spatial modality) couples.
2. Reduce ambient logic formula to temporal logic formula (CTL) by replacing spatial modalities with atomic propositions.
3. Generate state transition system of the ambient calculus specification with respect to reduction relations. This involves generation of initial state from given ambient calculus specification, generation of new states by applying available capabilities with respect to ambient calculus reduction relations and addition of new states to state transition system with transition relation.
4. Generate Kripke Structure from state transition system. This step involves the assignment of the values of the atomic propositions for each state of state transition system (labeling) by applying model checking for spatial modalities on ambient topology of the related state and the addition of a new state with its label (values of atomic propositions) to the Kripke Structure.
5. Generate NuSMV code from Kripke Structure and CTL Formula.

4.1 Ambient Topology and Spatial Formula Graphs

In [15], state information is represented with sets. In [6], calculus and logic information is represented as strings and algorithms are based on string operations. In the method proposed, ambient calculus specifications and logic formulas are represented as graphs. State information associated with a process specified in ambient calculus consists of static and dynamic properties. Static properties of state are the ambients and their hierarchical organization, i.e. the "ambient topology". The dynamic properties of the state are the capabilities and their dependencies on each other. Static and dynamic properties of an ambient calculus specification are kept in separate data structures.

Definition 1. *Ambient Topology, $G_{AT} = (N_{AT}, A_{AT})$, is an acyclic digraph where elements of set of nodes $v \in N_{AT}$ denotes ambients within the ambient calculus specification (elements of Λ) and arcs $a \in A_{AT}$, $a = \{xy \mid x, y \in N_{AT}\}$ denotes parent-child relation among ambients. The indegree of nodes $deg^{-}(v) = 1$ for any node (vertex) v whereas the outdegree of nodes $deg^{+}(v) \in \mathbb{N}$.*

The following defines capability trees which is a novel data structure used in our algorithm.

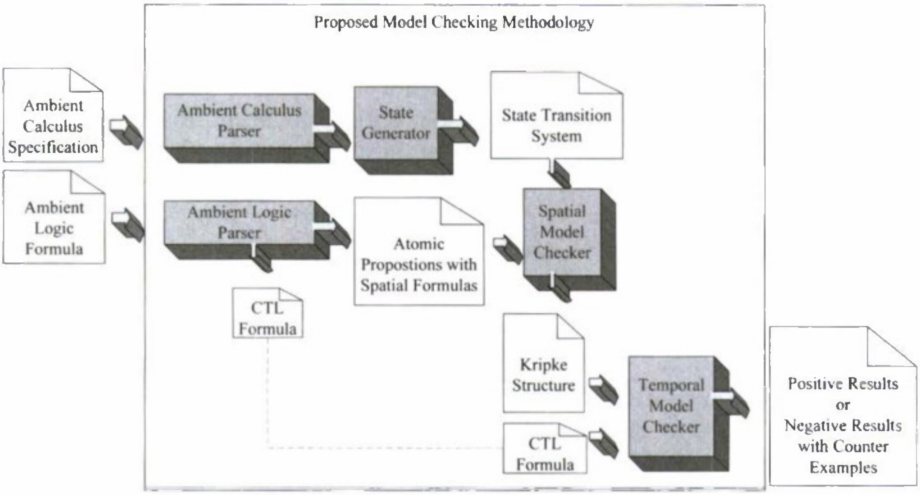


Fig. 1. Block diagram of the Ambient Calculus Model Checker

Definition 2. *Capability Tree, $G_{CT} = (N_{CT}, A_{CT})$, is an acyclic digraph where set of nodes $v \in N_{CT}$ denotes capabilities and arcs $a \in A_{CT}$, $a = \{xy \mid x,\ y \in N_{CT}\}$ denotes priority relation among capabilities. Nodes contain the information about which ambient the capability is attached and which ambient the capability effects. $deg^-(v) = 1$ for any node* v, *whereas $deg^+(v) \in \mathbb{N}$.*

Graphs representing formulas are more complex than the others. They are acyclic digraphs where nodes denote connectives and locations whereas arcs denote the operator-operand relation. There are multiple types of nodes and arcs in formula graphs because of the different structure of the ambient logic connectives.

Definition 3. *An ambient logic formula, $G_F = (N_F, A_F)$, is an acyclic digraph where*

- The set of nodes: $N_F = (N_L \cup N_{Binary} \cup N_{Unary} \cup N_{PC})$. N_L is the set of nodes representing ambients. Elements of N_L are labeled with elements of A. N_{Unary} is the set of nodes representing unary connectives ($\neg$, $\diamond$, $\Diamond$) at formulas. N_{Binary} is the set of nodes representing binary connectives, ($\vee$) at formulas. N_{PC} is the set of nodes representing parallel compositions at formulas.
- The set of arcs: $A_F = (A_{PC} \cup A_{Binary} \cup A_{Unary})$, where elements of A_{PC} represents parallel compositions, A_{Binary} represents binary connectives and A_{Unary} represents unary connectives of ambient logic formulas.
- $a_{pc} \in A_{PC} = x, y | x \in N_{PC},\ y \in (N_L \cup N_{Binary} \cup N_{Unary})$, $a_u \in A_{Unary} = (x, y \mid x \in (N_L \cup N_{Unary}), y \in N_{PC})$, $a_b \in A_{Binary} = (x, y | x \in N_{Binary},\ y \in N_{PC})$

- for $v \in N_F$, $deg^-(v) = 1$, for $v \in N_{PC}$, $deg^+(v) \in \mathbb{N}$, for $v \in N_{Unary}$, and $v \in N_L$, $deg^+(v) = 1$, for $v \in N_{Binary}$, $deg^+(v) = 2$.
- Elements of N_{PC} can have a special attribute to represent the **T** construct of the logic. If **T** attribute of a N_{PC} node is set to true this means the parallel composition of process that the N_{PC} node stands for, includes the constant **T**.

4.2 State Transition System Generation

In the proposed model checking methodology the state transition system is generated from the initial model specification by executing capabilities in the ambient calculus specification. Since replication is excluded from specifications, the state transition system can be represented by an acyclic digraph where nodes represent states and edges represent the execution of a capability. For selection of the next capability to execute, some condition checks are carried out. These conditions are the location of the object ambient and the availability of the subject ambient. A capability can not be executed if the location of the object ambient for the capability is not the current location, if it is prefixed by another capability path, or the parent ambient of the subject ambient is prefixed by a capability path. In the proposed method these conditions are checked each time a capability is to be executed.

In this work a new data structure is offered to represent temporal behaviors. The use of this data structure named "capability trees" eliminates the need to check the availability of a subject ambient. Capability paths are organized as an acyclic digraph that represent the interdependencies of capabilities. Capability trees are built at parsing stage so no pre-processing is needed. The selection of the next capability to execute starts from the root of this graph. This method guarantees that the capabilities of the parent processes are executed before the capabilities of child processes.

4.3 Checking Spatial Modalities

The basic element for building an ambient calculus model checker for ambient logic is to express and implement the satisfaction relation. In the proposed method, all the generated states generated must be checked against the spatial formulas. Ambient logic formulas are decomposed into a CTL formula and a set of spatial formulas by formula reduction. The ambient topology and the spatial formula graphs are inputs to the spatial model checker. The spatial model checking takes place before generation of Kripke Structures.

Matching of an ambient topology and a spatial formula is a recursive procedure in which ambient topology nodes are assigned to formula nodes. Matching process starts with assigning the ambient topology's root to the root of the spatial formula graph. Spatial formula nodes can forward the assigned ambient topology node to its children partially or completely in a recursive manner. Match process is successful when all nodes at ambient topology is matched to a spatial formula node. Match processes at different type of spatial formula nodes are different. Different match processes are introduced after auxiliary heuristic functions which are explained below.

Heuristic Functions. Heuristic functions are used at matching the Parallel composition (|) and Somewhere ($\diamond$) connectives. Former works try to match every alternative while searching a match for these connectives. In our proposed method, the number of these trials are reduced by the help of auxiliary heuristic functions. Some connectives of ambient logic called wildcard connectives match different kinds of ambient topology. These connectives are used for matching ambients of ambient topology which are not expressed in formulas. The constant **T** of the logic matches any ambient topology assigned to it. Negation connective of the logic can be seen as another kind of wildcard connective. Negations matches any ambient topology unless the sub formula of the negation matches this ambient topology. Another source of wildcard property is Somewhere connectives. The parallel process of the parent ambient are neglected when searching sublocations. So if the sublocation search is obtained by applying $\downarrow$ one or more times, the associated Somewhere connective gains a wildcard property. Function *wildcard* is a recursive function used for determining if a node of formula graphs has wildcard property.

It is not obvious to see which ambients are expected at sub formulas of Disjunction and Somewhere connectives. *guessExpectedAmbients* function is a recursive function which returns a set of expected ambient combinations for a formula graph node. The returned set includes all possible ambient combinations expected by children of that node. The returned value is a set instead of a single ambient combination. Function *findSublocation* is a recursive function used to find parent of an ambient at an ambient topology.

Matching of Spatial Formula. In a match between an ambient topology and spatial formula graph, all nodes of ambient topology must be matched with a node of spatial formula graph. Some nodes of spatial formula graphs can forward the ambient topology nodes assigned to them to their children, while others match assigned ambient topology nodes directly. The proposed spatial model checking algorithm tries alternative assignments of a given ambient topology nodes over a given spatial formula graph. The proposed spatial model checking algorithm is recursive where matching process starts from the roots of a graph and continues to underlying levels. If a suitable matching found at the upper level then matching process continues to find matches in lower levels. The match process is regulated by the semantics of spatial formula graph nodes.

4.4 Generation of Kripke Structure

A Kripke Structure is a state transition system where states are labeled by the set of atomic propositions which hold in that state. Atomic propositions can be considered as the marking of system properties.

Definition 4. *Let AP be a non-empty set of atomic propositions. A Kripke Structure is a four-tuple; M = (S, S0, R, L) where S is a finite set of states, S0*

$\subseteq S$ is the set of initial states, $R \subseteq S\times S$ is a transition relation, and $L: S \rightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions that are true in this state.

The state transition data structure provides sets S, S0 and relation R of a Kripke Structure. The elements of the set of atomic propositions come from formula reduction. In formula reduction, spatial formulas are replaced with atomic propositions. The function L is generated by applying spatial model checking for each state in state transition data structure against each spatial formula. Kripke Structure is obtained by attaching the values, coming from spatial model checking, into the state transition system graph.

4.5 NuSMV Code Generation

The model checking mechanism explained above provides CTL formulas and a Kripke Structure. The next step is the generation of NuSMV code which is semantically equivalent to the Kripke Structure and temporal logic formula. In the NuSMV specification a variable *state* is used for specifying states in the Kripke Structure. The other kind of variables used in NuSMV code generation is boolean variables for representing atomic propositions. CTL formulas provided by the formula reduction step are then converted to NuSMV code according to CTL formula graph provided by formula reduction, where the Sometime ($\Diamond$) connective is represented as EF and Everytime ($\Box$) connective is represented as AG. The atomic propositions are reflected into strings with their names.

4.6 Example for Spatial Model Checking Algorithm

Let's consider the scenario and policy example presented in Section 3.3. When the Ambient Calculus specification is input to the model checker, a total of 53 states are generated. One Atomic Proposition (AP) is generated, where

$$AP = \diamond\{\diamond \mathit{Host2}[\diamond \{\mathit{Data1}[T]|\mathit{Data2}[T]\}]|T\} \tag{4}$$

A part of the execution of the algorithm is presented in Table 3. Only the initial and the last two states are shown. For each state an action is executed to produce a new spatial state. For state 53 the spatial model checking algorithm matches the spatial formula AP to the current state of World.

Table 3. Part of output generated by the spatial model checker for the example policy presented in 3.3

State	Spatial state of World	AP	Action
0	Domain1|User1[]|Host1[File1|Data1]]]|| Domain2|Host2[User2[]|File2[Data2]]]]]	F	User2|out Host2|
52	Domain1 [User1 [Host1 [File1[]]]| Domain2 [Host2 [File2 [User2[]|Data2 []|Data1 [||]	F	User2[out File2]
53	Domain1 [User1 [Host1 [File1[]]]| Domain2 [Host2 [File2 [Data2 []|Data1 []]|User2[]]	T	-

4.7 Complexity and Performance Analysis

Time Complexity. Time complexity of generation states transition system is dependent on the number of capabilities. The execution of a capability causes a future state. In the worst case, all capabilities are independent. Independence of capabilities means that capabilities are in sequence or they operate on different ambients. Where n is the number of capabilities in the ambient calculus specification, the time complexity of generating state transition system in worst case is

$$\sum_{k=0}^{n} \frac{n!}{k!} \tag{5}$$

The time complexity of checking spatial modalities are dependent to the type and number of the connectives of the spatial formulas. The overall time cost of the match process for Somewhere connective is linear with the cost of match process of parallel composition for specifications with Somewhere connectives. However, the time complexity of the match process is exponential with the number of ambients as defined in Formula 6 where a_{ne} is the number of topmost ambients of the ambient topology which are not expected by the heuristic functions, d_w is the number of disjunctions which have wildcard property in the parallel composition, not is the number of negations in the parallel composition, sw_w is the number of Somewhere connectives which have wildcard property in the parallel composition:

$$O(a_{ne}^{(sw_w+not+d_w)}) \tag{6}$$

In contrast, when the brute force search is used for decomposing ambient calculus specifications, the time complexity is calculated as defined in Formula 7 where $a = a_{ne} + a_e$ is the total number of topmost ambients in the ambient topology, including those expected by the heuristic functions (a_e), l is the number of location in the parallel composition, sw is the number of Somewhere connectives which have not wildcard property in a parallel composition, d is the number of disjunctions which have not wildcard property in the parallel composition:

$$O((a)^{(sw_w+sw+l+not+d+d_w)}) \tag{7}$$

As presented above, the variables that effect the exponential complexity of the match process is significantly reduced by the proposed algorithm.

4.8 Space Complexity

Proposed algorithm builds a state transition system in a depth-first manner. The depth of the state transition system is at most equal to the number of capabilities. Therefore, the space complexity of the space generation is $O(n)$ where n is the number of capabilities. When checking spatial modalities, the space needed is equal to the size of the formula which is dependent on the number of connectives of the formula. Therefore, the space complexity of checking spatial modalities is $O(c)$, where c is the number of the connectives at formula.

4.9 Performance

Due to space limitations, the details of performance tests will not be presented. As a summary, our performance tests suggest that the state transition system generation cost outweighs the spatial model check for both time and space consumption. As an example to performance results, a specification with 16 ambients and 37 capabilities generates nearly 630,000 states with memory consumption under 8 MB and a time of under 300 seconds. The performance test has been run on an Intel G5 server with 2.93 GHz CPU and 10 GB memory.

5 Future Work and Conclusions

We presented a method and tool for the specification and verification of security policies of multi-domain mobile networks. The main focus of this method is location and mobility aspects of security policies. The basic elements of this method are predicate logic, ambient calculus and ambient logic. In this paper, model checking techniques are applied for verification of security policies and an ambient calculus model checker is presented.

The size of the state transition system is the most significant element at time and spatial cost of model checking. Number of states grows exponentially as capability number increase linearly. A partial order reduction might decrease the number of the states of the state transition system and reduce time consumption and size of generated NuSMV code. Investigating partial order reduction techniques for ambient calculus is a direction for our future work.

In our ongoing research we are developing tools for automatic extraction of formal process calculus specifications and logic formulas from security policy.In order to extract the Ambient Logic formula and specification from security policy, we are building a tool called "Formal Specification Generator". The tool will be based on analysis of scenarios depicting sequences of actions of system elements. These high-level actions are more suitable for our problem domain in contrast to Ambient Calculus primitives. Therefore our aim is to provide an automated means to translate high level policy and actions to formal calculus and logic specifications.

References

1. Becker, M., Fournet, C., Gordon, A.: Design and semantics of a decentralized authorization language. In: 20th IEEE Computer Security Foundations Symposium, pp. 3–15. IEEE Computer Society Press, Los Alamitos (2007)
2. Bertino, E., Ferrari, E., Buccafurri, F.: A logical framework for reasoning on data access control policies. In: 12th IEEE Computer Security Foundations Workshop, pp. 175–189. IEEE Computer Society Press, Los Alamitos (1999)
3. Cardelli, L., Gordon, A.D.: Anytime, anywhere: modal logics for mobile ambients. In: 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages - POPL 2000, pp. 365–377 (2000)

4. Cardelli, L., Gordon, A.D.: Mobile ambients. Theoretical Computer Science 240(1), 177–213 (2000)
5. Charatonik, W., Gordon, A., Talbot, J.: Finite-control mobile ambients. In: Le Métayer, D. (ed.) ESOP 2002. LNCS, vol. 2305, pp. 295–313. Springer, Heidelberg (2002)
6. Charatonik, W., Zilio, S.D., Gordon, A.D., Mukhopadhyay, S., Talbot, J.: Model checking mobile ambients. Theoretical Computer Science 308(1-3), 277–331 (2003)
7. Compagnoni, A., Bidinger, P.: Role-based access control for boxed ambients. Theoretical Computer Science 398(1-3), 203–216 (2008)
8. Cuppens, F., Saurel, C.: Specifying a security policy: a case study. In: 9th IEEE Computer Security Foundations Workshop, pp. 123–134. IEEE Computer Society Press, Los Alamitos (1996)
9. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder policy specification language. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 18–38. Springer, Heidelberg (2001)
10. Drouineaud, M., Bortin, M., Torrini, P., Sohr, K.: A first step towards formal verification of security policy properties for RBAC. In: Proc. QSIC (2004)
11. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Secur. 4(3), 224–274 (2001)
12. Giunchiglia, C.C., Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M.: Nusmv: a new symbolic model checker. International Journal on Software Tools for Technology Transfer 4, 410–425 (2000)
13. Jajodia, S., Samarati, P., Subrahmanian, V.S.: A logical language for expressing authorizations. In: IEEE Symposium on Security and Privacy, pp. 31–42 (1997)
14. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible support for multiple access control policies. ACM Trans. Database Syst. 26(2), 214–260 (2001)
15. Mardare, R., Priami, C., Quaglia, P., Vagin, O.: Model checking biological systems described using ambient calculus. Computational Methods in Systems Biology, 85–103 (2005)
16. Ryutov, T., Neuman, C.: Representation and evaluation of security policies for distributed system services. In: DARPA Information Survivability Conference and Exposition, pp. 172–183 (2000)
17. Scott, D.: Abstracting application-level security policy for ubiquitous computing. Ph.D. thesis, University of Cambridge (2005)
18. Sohr, K., Drouineaud, M., Ahn, G., Gogolla, M.: Analyzing and managing Role-Based access control policies. IEEE Transactions on Knowledge and Data Engineering 20(7), 924–939 (2008)
19. Unal, D., Caglayan, M.U.: Theorem proving for modeling and conflict checking of authorization policies. In: Proc. ISCN (2006)
20. Woo, T.Y.C., Lam, S.S.: Authorizations in distributed systems: A new approach. Journal of Computer Security 2, 107–136 (1993)
21. Zhang, N., Guelev, D., Ryan, M.: Synthesising verified access control systems through model checking. Journal of Computer Security 16(1), 1–61 (2007)

Credentials Management for High-Value Transactions

Glenn Benson[1], Shiu-Kai Chin[2], Sean Croston[1],
Karthick Jayaraman[2], and Susan Older[2]

[1] JP Morgan Chase
{glenn.benson,sean.b.croston}@jpmchase.com
[2] EECS Department, Syracuse University, Syracuse, New York 13244
{skchin,kjayaram,sbolder}@syr.edu

Abstract. Partner key management (PKM) is an interoperable credential management protocol for online commercial transactions of high value. PKM reinterprets traditional public key infrastructure (PKI) for use in high-value commercial transactions, which require additional controls on the use of credentials for authentication and authorization. The need for additional controls is met by the use of partner key practice statements (PKPS), which are machine-readable policy statements precisely specifying a bank's policy for accepting and processing payment requests. As assurance is crucial for high-value transactions, we use an access-control logic to: (1) describe the protocol, (2) assure the logical consistency of the operations, and (3) to make the trust assumptions explicit.

Keywords: authentication, authorization, protocols, trust, logic.

1 Introduction

Authorizing online high-value commercial transactions requires a higher level of diligence when compared to consumer or retail transactions. A single high-value transaction may involve the transfer of hundreds of millions of dollars. The inherent risk associated with wholesale online banking compels many banks to require additional security beyond authenticating users at login time. Additional security often takes the form of tighter controls and limits on the use of credentials. Ultimately, each bank trusts itself more than any other entity. This naturally leads to the practice of banks issuing their own credentials. Historically, a cash manager of a corporation would hold separate credentials from each bank with which he or she deals. While this serves the needs of commercial banks, as corporations want to simultaneously hold accounts in multiple banks, the insistence upon and proliferation of unique credentials is viewed by customers as poor service. Hence, it is increasingly important for global financial services providers, such as JP Morgan Chase, to offer credentials that: (1) are interoperable to provide customer convenience, and (2) meet the needs of high-value commercial transactions in terms of authentication, authorization, and liability.

Traditional Public Key Infrastructure (PKI) credentials while interoperable, alone are insufficient to surmount the following obstacles inherent to the use of interoperable credentials in high-value transactions:

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 169–182, 2010.

1. *Autonomy*: Interoperability and autonomy are in tension with each other. An implication of interoperability is the need to allow audits. For example, say Second Bank is contemplating recognizing credentials issued by First Bank. Second Bank would understandably want to audit First Bank's practices as a credential issuer against Second Bank's policies. Understandably, First Bank would be reluctant to agree to audits of its operations by competitors such as Second Bank.
2. *Liability*: Non-bank issuers of PKI credentials neither want, nor are in a position to accept, liability for failed high-value transactions. One way around this is for a bank to issue its own credentials to limit risk and to recognize only the credentials it issues; however, the solution is not interoperable by definition.
3. *Expense*: If commercial banks were to recognize non-bank certificate issuers for high-value commercial transactions, then commercial banks would need to be connected to the non-bank certificate issuers. This is an added operational expense for banks, which is another barrier to achieving interoperability.

In this paper, we describe an interoperable certificate management protocol called *partner key management* (PKM). PKM is designed to address the three obstacles to interoperability of credentials in high-value transactions described above. Under the PKM model, each bank publishes a *partner key practice statement* (PKPS), which is a machine readable document that describes the bank's policy for accepting interoperable credentials. PKM enables each bank to avoid liability on transactions executed at any other bank, while preserving credential interoperability. Furthermore, PKM supports a general validation model, where each corporation need only connect to the credential issuers to which it subscribes. Moreover, we describe the certificate management protocol using an access-control logic to prove its logical consistency and also to make the underlying trust assumptions explicit.

The rest of this paper is organized as follows. Section 2 presents the PKM model, PKPS, and sender validation. Section 3 defines the syntax, semantics, and inference rules of the logic used to describe and reason about PKM. Section 4 is an overview of how key parts of PKPS are expressed in the logic. Section 5 provides an extended example describing and analyzing the operation of PKM. Related work is briefly discussed in Section 6. We offer conclusions in Section 7.

2 Partner Key Management

2.1 Credentials Registration

The PKM model focuses on authorization to use a credential as opposed to secure distribution of a credential. As an analogy, consider mobile phone distribution logistics. A user may purchase a mobile phone from any distributor. At the time that the user physically acquires the phone, the telecom operator does not know the user's identity and does not allow use of the phone. Subsequently, the user and the telecom operator agree to terms of use; and the mobile phone operator authorizes the phone's connection to the telecom network. In the PKM model, the credential plays the role of the phone, and the bank plays a similar role to the telecom operator.

In PKM, the user first obtains a credential from a credential distributor. The credential distributor has the responsibility to distribute 'secure' credentials under a definition of security defined by the operator. For example, one operator may only distribute certificates on secured USB devices, while another operator may distribute software for self-signed certificates. After obtaining a credential, the user submits a request to each of his or her banks to allow use of the credential. On this step, the bank has two responsibilities. First, the bank must securely assure itself of the user's true identity. Second, the bank must examine the credential to determine if the credential meets the bank's standards. For example, some banks may prohibit credentials other than certificates that reside in a secured hardware token. If the bank accepts the credential, then the bank authorizes the credential to represent the user. The user may use the same credential with multiple banks by appropriately registering the credential with the respective banks. The authorization process may vary between the banks. Each bank may have its own operational policy governing the conditions in which it accepts the credentials based upon the bank's published operating rules.

In effect, the credentials are interoperable, and banks have the liberty to follow their own procedure for accepting the credentials and allowing users to employ those credentials. The result is an infrastructure that allows the possibility of interoperability without mandating interoperability. If two banks agree to accept a single credential, then that credential would interoperate between the two banks. No bank needs to rely upon any other bank or external credential provider.

2.2 Partner Key Practice Statement

Banks participating in the PKM model publish an XML document called the *Partner Key Practice Statement* (PKPS), which is written using WS-Policy [1]. A PKPS defines how a corporation and a bank agree to work together, as governed by their mutually agreed upon security procedures. The corporation and the bank have the freedom to impose almost any conditions to which they mutually agree, provided that the conditions do not require unsupportable programming logic. The list below presents some examples types of information that may appear in a PKPS:

1. *Credential Media:* The definition of the credential media may mandate a smart card, USB token, HSM, FIPS-140-2, or a software credential.
2. *Credential Provider:* This item contains the list of credential providers to which the corporation and the bank mutually subscribe. Example providers are third party trusted providers, self-signed certificates, the corporation's, or the bank's own infrastructure.
3. *Revocation:* The revocation definition describes the type of permissible credential revocation mechanism, e.g., certificate revocation list (CRL), online certificate status protocol (OCSP) [2], etc. The revocation definition also describes the party responsible for enforcing credential revocation; and it describes any specific usage practice. For example, the revocation mechanism may mandate that the recipient of a signature validate a CRL signed by a particular party.
4. *Timestamp:* The timestamp definition defines timestamp rules and the timestamp provider, if any. The timestamp definition may specify a real-time

threshold value. The recipient must ensure that it receives and validates a signature before the threshold timelimit after the timestamp. For example, a six hour threshold value means that the recipient must validate a signature before six hours expires after the timestamp.
5. *Signature Policy:* The PKPS can specify the number of signatures required for a specific type of transaction, and the roles of signatories. An example of a signature policy is one which requires both an individual signature and a corporate "system" signature in order to consider either signature as valid.
6. *Credential Technology:* A certificate that supports the X.509 standard is an obvious choice for interoperability. However, additional technologies such as the portable security transaction protocol (PSTP) [3] exist, and the PKPS may specify alternative technologies.

The security requirements mutually agreed to by the bank and the corporation are reflected in a specific PKPS, or possibly a list of PKPSs. The security requirements may mandate that the corporation must attach the PKPS on each signed transaction in order to consider any signature valid.

2.3 Revocation

This paper presents three example validation models. A bank's PKSP should define the model that a particular bank allows.

1. **Receiver validation:** The receiver validation model is typically used in a PKI model. First, Alice submits a signed transaction to the bank. Upon receipt, the bank validates Alice's signature against a CRL or OCSP responder managed by the certificate provider.
2. **Sender validation without evidence:** Alice submits signed transactions to the bank, but the bank performs no revocation check. Alice's company and the bank manage Alice's credential using mechanism outside the scope of the signed transaction.
3. **Sender validation with evidence:** Alice submits her certificate to an OCSP responder, and obtains a response signed by the OCSP responder. Alice signs the transaction and the OCSP response, and then submits to the bank. The bank validates both Alice's signature and the OCSP responder's signature. If the bank finds no error, then the bank accepts the transaction.

Each bank has the opportunity to allow any of the three example models, or build its own variant model. Multiple banks may all accept the same credential from Alice, while requiring different revocation models.The second model, sender validation without evidence, merits further discussion. If Alice proves to be an untrustworthy person, then Alice's company reserves the right to disable Alice's credential. For example, if Alice has a gambling problem, then authorized representatives of Alice's company should contact each of its banks with the instruction to stop allowing Alice's credential. Another use case which also results in credential disabling, is one where Alice contacts each bank because she suspects that her own credential was lost or stolen.

An OCSP responder, or a certificate revocation list is merely a revocation mechanism optimized for scalability. As opposed to requiring the Alice's company to contact each of its banks, an OCSP responder or Certificate Revocation

List provides a centralized repository which handles certificate revocation. The advantage of the OCSP responder or certificate revocation list is scalability as opposed to security. If Alice were authorized to transact on accounts at hundreds or thousands of banks, then the second model (sender validation without evidence) would not be practical. However, in practice, wholesale banking does not need such enormous scalability. Rather, Alice typically works with just a handful of banks. Although Alice's company may find the credential disabling process to be relatively tedious because the company needs to contact each of the banks in the handful, we normally find that corporations employ the credential disabling process relatively infrequently.

In practice, corporations tend to contact each of their banks whenever a user's credential changes status, even if the bank happens to use the traditional receiver validation model. In fact, some banks require immediate notification of such events in their operating model. Intuitively, if the corporation ceases to trust Alice to authorize high-value transactions, then the corporation probably wants to contact each of its banks directly.

Both the second and the third models assume sender validation, as opposed to receiver validation. An advantage of sender validation is that it better handles expense. Suppose, for example, a corporation agrees to the services of a new credential distributor. Credential interoperability encourages a dynamic market by allowing the corporation the freedom to choose any acceptable credential distributor. In the receiver validation model, the corporation could not use that credential with its bank until the bank agrees to build an online connection to the credential distributor's OCSP responder or certificate revocation list. In the sender validation models, on the other hand, the corporation may immediately use the credential with the bank without waiting for the costly and possibly slow technology development process.

3 An Access-Control Logic and Calculus

We use an access-control logic to describe and reason about the validity of acting on payment instructions. This section introduces the syntax, semantics, and inference rules of the logic we use.

3.1 Syntax

Principal Expressions. Let P and Q range over a collection of principal expressions. Let A range over a countable set of simple principal names. The abstract syntax of principal expressions is:

$$P ::= A \ / \ P\&Q \ / \ P \mid Q$$

The principal $P\&Q$ ("P in conjunction with Q") is an abstract principal making exactly those statements made by both P and Q; $P \mid Q$ ("P quoting Q") is an abstract principal corresponding to principal P quoting principal Q.

Access Control Statements. The abstract syntax of statements (ranged over by φ) is defined as follows, where P and Q range over principal expressions and p ranges over a countable set of *propositional variables*:

$$\begin{gathered}\varphi ::= p \ / \ \neg\varphi \ / \ \varphi_1 \wedge \varphi_2 \ / \ \varphi_1 \vee \varphi_2 \ / \ \varphi_1 \supset \varphi_2 \ / \ \varphi_1 \equiv \varphi_2 \ / \\ P \Rightarrow Q \ / \ P \textsf{ says } \varphi \ / \ P \textsf{ controls } \varphi \ / \ P \ \textsf{ reps } Q \textsf{ on } \varphi\end{gathered}$$

Informally, a formula $P \Rightarrow Q$ (pronounced "P speaks for Q") indicates that *every* statement made by P can also be viewed as a statement from Q. A formula P controls φ is syntactic sugar for the implication (P says φ) $\supset \varphi$: in effect, P is a trusted authority with respect to the statement φ. P reps Q on φ denotes that P is Q's delegate on φ; it is syntactic sugar for (P says (Q says φ)) $\supset$ Q says φ. Notice that the definition of P reps Q on φ is a special case of controls and in effect asserts that P is a trusted authority with respect to Q saying φ.

3.2 Semantics

Kripke structures define the semantics of formulas.

Definition 1. *A* Kripke structure $\mathcal{M}$ *is a three-tuple* $\langle W, I, J\rangle$, *where:*

- W *is a nonempty set, whose elements are called* worlds.
- I : ***PropVar*** $\rightarrow \mathcal{P}(W)$ *is an* interpretation *function that maps each propositional variable p to a set of worlds.*
- J : ***PName*** $\rightarrow \mathcal{P}(W \times W)$ *is a function that maps each principal name A to a relation on worlds (i.e., a subset of* $W \times W$*).*

We extend J to work over arbitrary *principal expressions* using set union and relational composition as follows:

$$J(P\&Q) = J(P) \cup J(Q)$$
$$J(P \mid Q) = J(P) \circ J(Q),$$

where

$$J(P) \circ J(Q) = \{(w_1, w_2) \mid \exists w'.(w_1, w') \in J(P) \text{ and } (w', w_2) \in J(Q)\}$$

Definition 2. *Each Kripke structure* $\mathcal{M} = \langle W, I, J\rangle$ *gives rise to a function*

$$\mathcal{E}_{\mathcal{M}}[\![-]\!] : \textbf{\textit{Form}} \rightarrow \mathcal{P}(W),$$

where $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ *is the set of worlds in which* φ *is considered true.* $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ *is defined inductively on the structure of* φ, *as shown in Figure 1.*

Note that, in the definition of $\mathcal{E}_{\mathcal{M}}[\![P \text{ says } \varphi]\!]$, $J(P)(w)$ *is simply the image of world* w *under the relation* $J(P)$.

3.3 Inference Rules

In practice, relying on the Kripke semantics alone to reason about policies and behavior is inconvenient. Instead, inference rules are used to manipulate formulas in the logic. All logical rules must be sound to maintain consistency.

Definition 3. *A rule of form* $\dfrac{H_1 \cdots H_n}{C}$ *is sound if for all Kripke structures* $\mathcal{M} = \langle W, I, J\rangle$, *if* $\mathcal{E}_{\mathcal{M}}[\![H_i]\!] = W$ *for each* $i \in \{1, \ldots, n\}$, *then* $\mathcal{E}_{\mathcal{M}}[\![C]\!] = W$.

The rules in Figures 2 and 3 are all sound. If sound rules are used throughout, then the conclusions derived using the inference rules are sound, too.

$$
\begin{aligned}
\mathcal{E}_{\mathcal{M}}[\![p]\!] &= I(p) \\
\mathcal{E}_{\mathcal{M}}[\![\neg\varphi]\!] &= W - \mathcal{E}_{\mathcal{M}}[\![\varphi]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \wedge \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \vee \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] &= (W - \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!]) \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \equiv \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2 \supset \varphi_1]\!] \\
\mathcal{E}_{\mathcal{M}}[\![P \Rightarrow Q]\!] &= \begin{cases} W, & \text{if } J(Q) \subseteq J(P) \\ \emptyset, & \text{otherwise} \end{cases} \\
\mathcal{E}_{\mathcal{M}}[\![P \textsf{ says } \varphi]\!] &= \{w \mid J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\} \\
\mathcal{E}_{\mathcal{M}}[\![P \textsf{ controls } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![(P \textsf{ says } \varphi) \supset \varphi]\!] \\
\mathcal{E}_{\mathcal{M}}[\![P \ \textsf{ reps } Q \textsf{ on } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![P \mid Q \textsf{ says } \varphi \supset Q \textsf{ says } \varphi]\!]
\end{aligned}
$$

Fig. 1. Semantics

$$\textit{Taut} \quad \frac{}{\varphi} \quad \text{if } \varphi \text{ is an instance of a prop-logic tautology}$$

$$\textit{Modus Ponens} \quad \frac{\varphi \qquad \varphi \supset \varphi'}{\varphi'} \qquad\qquad \textit{Says} \quad \frac{\varphi}{P \textsf{ says } \varphi}$$

$$\textit{MP Says} \quad \frac{}{(P \textsf{ says } (\varphi \supset \varphi')) \supset (P \textsf{ says } \varphi \supset P \textsf{ says } \varphi')}$$

$$\textit{Speaks For} \quad \frac{}{P \Rightarrow Q \supset (P \textsf{ says } \varphi \supset Q \textsf{ says } \varphi)}$$

$$\textit{Quoting} \quad \frac{}{P \mid Q \textsf{ says } \varphi \equiv P \textsf{ says } Q \textsf{ says } \varphi}$$

$$\textit{\&Says} \quad \frac{}{P\&Q \textsf{ says } \varphi \equiv P \textsf{ says } \varphi \wedge Q \textsf{ says } \varphi}$$

$$\textit{Idempotency of} \Rightarrow \quad \frac{}{P \Rightarrow P} \qquad\qquad \textit{Monotonicity of} \mid \quad \frac{P' \Rightarrow P \quad Q' \Rightarrow Q}{P' \mid Q' \Rightarrow P \mid Q}$$

$$\textit{Associativity of} \mid \quad \frac{P \mid (Q \mid R) \textsf{ says } \varphi}{(P \mid Q) \mid R \textsf{ says } \varphi}$$

$$P \textsf{ controls } \varphi \quad \stackrel{\text{def}}{=} \quad (P \textsf{ says } \varphi) \supset \varphi$$

$$P \ \textsf{ reps } Q \textsf{ on } \varphi \stackrel{\text{def}}{=} P \mid Q \textsf{ says } \varphi \supset Q \textsf{ says } \varphi$$

Fig. 2. Core Inference Rules

4 Expressing Statements and the PKPS in Logic

With the definition of the syntax and semantics of access-control logic, we provide an introduction to expressing actual payment instructions and the PKPS in logic.

Statements and Certificates: Statements and requests are made by principals. Requests are logical statements. For example, say Alice wants to transfer $\$10^6$ dollars from $acct_1$ to $acct_2$. If $\langle transfer\ 10^6, acct_1, acct_2\rangle$ denotes the proposition *it is justifiable to transfer* $\$10^6$ *from* $acct_1$ *to* $acct_2$, then we can represent Alice's request as $Alice$ says $\langle transfer\ 10^6, acct_1, acct_2\rangle$. Credentials or certificates are statements, usually signed with a cryptographic key. For example, assume we

$$\textit{Quoting (1)}\ \frac{P \mid Q \textsf{ says } \varphi}{P \textsf{ says } Q \textsf{ says } \varphi} \qquad \textit{Quoting (2)}\ \frac{P \textsf{ says } Q \textsf{ says } \varphi}{P \mid Q \textsf{ says } \varphi}$$

$$\textit{Controls}\ \frac{P \textsf{ controls } \varphi \quad P \textsf{ says } \varphi}{\varphi} \qquad \textit{Derived Speaks For}\ \frac{P \Rightarrow Q \quad P \textsf{ says } \varphi}{Q \textsf{ says } \varphi}$$

$$\textit{Reps}\ \frac{Q \textsf{ controls } \varphi \quad P \ \textsf{ reps } Q \textsf{ on } \varphi \quad P \mid Q \textsf{ says } \varphi}{\varphi}$$

$$\textit{Rep Says}\ \frac{P \ \textsf{ reps } Q \textsf{ on } \varphi \quad P \mid Q \textsf{ says } \varphi}{Q \textsf{ says } \varphi}$$

Fig. 3. Derived Rules Used in this Paper

believe K_{CA} is the key used by certificate authority CA. With this belief, we would interpret a statement made by K_{CA} to come from CA. In particular, if K_{CA} says $(K_{Alice} \Rightarrow Alice)$, we would interpret this public key certificate signed by K_{CA} as having come from CA.

Authority and Jurisdiction: Jurisdiction statements identify who or what has authority, specific privileges, powers, or rights. In the logic, jurisdiction statements usually are controls statements. For example, if Alice has the right to transfer a \$$10^6$ dollars from $acct_1$ to $acct_2$, we say $Alice$ controls $\langle transfer\ \ 10^6, acct_1, acct_2\rangle$. If Alice has jurisdiction on $\langle transfer\ 10^6, acct_1, acct_2\rangle$ and Alice requests $\langle transfer\ 10^6, acct_1, acct_2\rangle$, then the *Controls* inference rule in Figure 3 allows us to infer the soundness of $\langle transfer\ 10^6, acct_1, acct_2\rangle$.

$$\frac{Alice \textsf{ controls } \langle transfer\ 10^6, acct_1, acct_2\rangle \qquad Alice \textsf{ says } \langle transfer\ 10^6, acct_1, acct_2\rangle}{\langle transfer\ 10^6, acct_1, acct_2\rangle.}$$

Proxies and delegates. Often, something or somebody makes the requests to the guards protecting the resource on behalf of the actual principals, who are the sources of the requests. In an electronic transaction, a cryptographic key is used as a proxy for a principal. Recall that K_{CA} says $(K_{Alice} \Rightarrow Alice)$ is a public key certificate signed with the public key K_{CA} of the certification authority. The certification authority's key, K_{CA}, is installed on the computer using a trustworthy key distribution process, and the trust in the key is captured using the statement $K_{CA} \Rightarrow CA$. If we get a certificate signed using K_{CA}, then we would attribute the information in that certificate to CA. For example, using the *Derived Speaks For* rule in Figure 3 we can conclude that certificate authority CA vouches for K_{Alice} being Alice's public key:

$$\frac{K_{CA} \Rightarrow CA \qquad K_{CA} \textsf{ says } (K_{Alice} \Rightarrow Alice)}{CA \textsf{ says } (K_{Alice} \Rightarrow Alice).}$$

$K_{Alice} \Rightarrow Alice$ is a statement of trust on K_{Alice}, where all statements made by K_{Alice} are attributed to Alice. However, in some situations, a principal may be trusted only on specific statements. For example, K_{Alice} may be trusted on a statement requesting a transfer of a million dollars. However, K_{Alice} may not be trusted on a statement $K_{Bob} \Rightarrow Bob$. This notion of a constrained

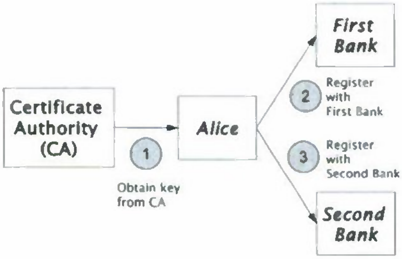


Fig. 4. Partner key management

delegation, where a principal's delegate is trusted on specific statements, is described using reps formulas. For example, if K_{Alice} is trusted to be Alice's delegate on the statement $\langle transfer\ 10^6, acct_1, acct_2\rangle$, we would write: K_{Alice} reps $Alice$ on $\langle transfer\ 10^6, acct_1, acct_2\rangle$.

From the semantics of reps, if we recognize K_{Alice} as Alice's delegate, in effect we are saying that K_{Alice} is trusted on Alice stating that she wishes a million dollars to be transferred from $acct_1$ to $acct_2$. If K_{Alice} says Alice says transfer a million dollars from $acct_1$ to $acct_2$, we will conclude that Alice has made the request. Using the *Rep Says* rule in Figure 3 we can conclude:

$$\frac{\begin{array}{c} K_{Alice} \text{ reps } Alice \text{ on } \langle transfer\ 10^6, acct_1, acct_2\rangle \\ K_{Alice} \mid Alice \text{ says } \langle transfer\ 10^6, acct_1, acct_2\rangle \end{array}}{Alice \text{ says } \langle transfer\ 10^6, acct_1, acct_2\rangle.}$$

5 An Extended Example

In this section, we illustrate PKM with a hypothetical example. Suppose Alice is a cash manager who works for the Widget Corporation. Further suppose that Widget uses three banks: First, Second, and Third Bank. Suppose the three banks use different procedures for authorizing credentials, which the Widget corporate Treasurer finds acceptable. Both First and Second Banks use the PKM model, while for explanatory purposes only, assume that Third Bank uses the PKI model. Both First and Second Bank allow Alice to obtain a credential from any provider, while Third Bank requires Alice to obtain a credential from a specific certificate authority that we will refer to as (CA). Therefore, Alice obtains a certificate from CA that can be used with all the Three banks. Because First and Second Banks use PKM, Alice registers the certificate with both the banks. First and Second Bank describe their procedure for accepting certificates in a partner key practice statement (PKPS). Both First Bank and Second Bank require Alice to submit a signed PKPS along with each transaction. First Bank requires Widget to check for revocation prior to Alice sending the payment instruction. There is a mutual agreement of sender liability if Widget does not check for revocation before affixing the signature. Second Bank requires Alice to sign an OCSP response obtained from the certificate provider, and Second Bank will validate Alice's certificate using the OCSP response. Third Bank uses

Payment Instruction:
1. K_{Alice} says $\langle transfer\ 10^6, acct_1, acct_2\rangle$
2. K_{Alice} says $\langle First\ Bank\ PKPS, timestamp\rangle$

Entitlement:
1. $Alice$ controls $\langle transfer\ 10^6, acct_1, acct_2\rangle$

Mutually Agreed Operational Rules:
1. $First$ controls $(K_{Alice} \Rightarrow Alice)$
2. K_{Alice} says $\langle First\ Bank\ PKPS, timestamp\rangle$ $\supset \langle K_{Alice} Validated, timestamp\rangle$
3. $\langle K_{Alice} Validated, timestamp\rangle$ $\supset (First$ says $K_{Alice} \Rightarrow Alice)$

Fig. 5. First Bank: Payment instruction, entitlement, and operating rules

```
<pkps:pkps id = First>
  <wsp:policyattachment>
     <wsp:appliesto>
         <pkps:requester>
             <pkps:any/>
         </pkps:requester>
         <pkps:receiver>
             First
          </pkps:receiver>
      </wsp:appliesto>
      <wsp:policy>
         <wsp:all>
            <pkps:validation-model>
                <pkps:sender-no-evidence/>
            </pkps:validation-model>
         </wsp:all>
      </wsp:policy>
  </wsp:policyattachment>
</pkps:pkps>
```

Fig. 6. First Bank's PKPS

the traditional PKI model, so there is no PKPS involved. Also, Third Bank uses a receiver validation model, so Third Bank will connect to the CA's OCSP responder to validate the certificates.

We will use the access-control logic (Section 3) to describe in detail the operations of the three banks for a hypothetical transaction, in which Alice requests a transfer for $\$10^6$ from Widget's account to a different account. For each bank, we provide a derived inference for justifying the bank's decision to act on the payment instruction. The proof of these derived inference rules are a direct application of the inference rules described in Section 3.3. Our objective is to primarily show the differences between PKI and PKM with respect to how the credentials are managed. We use the access-control logic to show the logical consistency of the operations and also to make the mutually agreed operating rules explicit.
Important note: In the hypothetical example, Alice requires an entitlement to request a transaction. The methods commonly used by banks to issue such entitlements to Alice are outside the scope of this paper. For the purpose of our illustration, we will assume that Alice has the necessary entitlement.

5.1 First Bank

Figure 5 contains an example payment instruction for First Bank. The payment instruction comprises two statements, (1) a statement signed using K_{Alice} requesting transfer of \$1 million, (2) First's PKPS (Figure 6) and timestamp signed using K_{Alice}. As per the mutually agreed operational rules, First has the authority for authorizing Alice to use K_{Alice}, and First issues such an authorization when K_{Alice} is validated. According to First's PKPS, the sender is expected to validate K_{Alice} prior to the transaction, and First assumes that the K_{Alice} is validated appropriately when Alice signs First's PKPS with K_{Alice}. The following derived inference rule justifies the bank's decision to act on the payment instruction.

Payment Instruction:

1. K_{Alice} says $\langle transfer\ 10^6, acct_1, acct_2\rangle$
2. $(K_{Alice} \mid K_{CA})$ says $\langle K_{Alice} Validated, timestamp\rangle$
3. K_{Alice} says $\langle Second\ Bank\ PKPS, timestamp\rangle$

Entitlement:

1. $Alice$ controls $\langle transfer\ 10^6, acct_1, acct_2\rangle$

Mutually Agreed Operational Rules

1. $Second$ controls $K_{Alice} \Rightarrow Alice$
2. $K_{CA} \Rightarrow CA$
3. K_{Alice} says $\langle Second\ Bank\ PKPS, timestamp\rangle \supset$ CA controls $\langle K_{Alice} Validated, timestamp\rangle \wedge$ K_{Alice} reps K_{CA} on $\langle K_{Alice} Validated, timestamp\rangle$
4. $\langle K_{Alice} Validated, timestamp\rangle \supset$ $Second$ says $K_{Alice} \Rightarrow Alice$

Fig. 7. Second Bank: Payment instruction, entitlement, and operating rules

```
<pkps:pkps id=Second>
 <wsp:policyattachment>
   <wsp:appliesto>
      <pkps:requester>
         <pkps:any/>
      </pkps:requester>
      <pkps:receiver>
        Second
       </pkps:receiver>
    </wsp:appliesto>
    <wsp:policy>
      <wsp:all>
        <pkps:revocation>
           <pkps:sender-with-evidence/>
        </pkps:revocation>
      </wsp:all>
    </wsp:policy>
 </wsp:policyattachment>
</pkps:pkps>
```

Fig. 8. Second Bank's PKPS

$$\textit{First Bank}\quad \frac{\begin{array}{c} K_{Alice} \text{ says } \langle transfer\ 10^6, acct_1, acct_2\rangle \\ K_{Alice} \text{ says } \langle First\ Bank\ PKPS, timestamp\rangle \\ Alice \text{ controls } \langle transfer\ 10^6, acct_1, acct_2\rangle \\ First \text{ controls } K_{Alice} \Rightarrow Alice \\ K_{Alice} \text{ says } \langle First\ Bank\ PKPS, timestamp\rangle \supset \langle K_{Alice} Validated, timestamp\rangle \\ \langle K_{Alice} Validated, timestamp\rangle \supset \langle First \text{ says } K_{Alice} \Rightarrow Alice\rangle \end{array}}{\langle transfer\ 10^6, acct_1, acct_2\rangle}$$

5.2 Second Bank

The payment instruction for Second Bank, in Figure 7, comprises three statements, (1) a statement signed using K_{Alice} requesting transfer of \$1 million, (2) CA's OCSP response for K_{Alice} signed using K_{Alice}, (3) PKPS (Figure 8) and timestamp signed using K_{Alice}. Second Bank has authority for authorizing Alice to use K_{Alice}, similar to the First Bank, but uses the sender-validation-with-evidence model for validation. When K_{Alice} signs Second's PKPS, both parties agree to two operating rules for validating K_{Alice}. First, CA has authority for validating K_{Alice}. Second, K_{Alice} is a recognized delegate of K_{CA} for relaying the OCSP response for K_{Alice}. The following derived inference rule justifies the bank's decision to act on the payment instruction.

$$\textit{Second Bank}\quad \frac{\begin{array}{c} K_{Alice} \text{ says } \langle transfer\ 10^6, acct_1, acct_2\rangle \\ (K_{Alice} \mid K_{CA}) \text{ says } \langle K_{Alice} Validated, timestamp\rangle \\ K_{Alice} \text{ says } \langle Second\ Bank\ PKPS, timestamp\rangle \\ Alice \text{ controls } \langle transfer\ 10^6, acct_1, acct_2\rangle \\ Second \text{ controls } K_{Alice} \Rightarrow Alice \\ K_{CA} \Rightarrow CA \\ K_{Alice} \text{ says } \langle Second\ Bank\ PKPS, timestamp\rangle \supset \\ \{CA \text{ controls } \langle K_{Alice} Validated, timestamp\rangle \wedge \\ K_{Alice} \text{ reps } K_{CA} \text{ on } \langle K_{Alice} Validated, timestamp\rangle\} \\ \langle K_{Alice} Validated, timestamp\rangle \supset Second \text{ says } K_{Alice} \Rightarrow Alice \end{array}}{\langle transfer\ 10^6, acct_1, acct_2\rangle}$$

5.3 Third Bank

The payment instruction for Third Bank, in Figure 9, is a statement signed using K_{Alice} for requesting a transfer of \$1 million. Third Bank believes in the jurisdiction of the CA for identifying the Key of Alice. When Third Bank receives the public key certificate for K_{Alice}, it validates it by connecting to CA's OCSP responder. On successful validation, Third Bank is convinced that K_{Alice} belongs to Alice. For the sake of brevity, we do not describe the actual validation process in the logic. Moreover, doing so does not change the trust assumptions, more specifically does not affect Third Bank's belief in CA's authority. The following derived inference rule justifies the bank's decision to act on the payment instruction.

$$\textit{Third Bank}\quad \frac{\begin{array}{c} K_{Alice}\ \textsf{says}\ \langle transfer\ \$10^6, acct_1\rangle \\ Alice\ \textsf{controls}\ \langle transfer\ \$10^6, acct_1\rangle \\ K_{CA}\ \textsf{says}\ K_{Alice} \Rightarrow Alice \\ K_{CA} \Rightarrow CA \\ CA\ \textsf{controls}\ K_{Alice} \Rightarrow Alice \end{array}}{\langle transfer\ 10^6, acct_1, acct_2\rangle}$$

Payment Instruction:
1. K_{Alice} says $\langle transfer\ 10^6, acct_1, acct_2\rangle$

Entitlement
1. $Alice$ controls $\langle transfer\ \$10^6, acct_1\rangle$

Public Key Certificate
1. K_{CA} says $K_{Alice} \Rightarrow Alice$

Trust Assumptions:
1. $K_{CA} \Rightarrow CA$
2. CA controls $K_{Alice} \Rightarrow Alice$

Fig. 9. Third Bank: Payment instruction, entitlement, certificates, and trust assumptions

5.4 Analysis

The traditional PKI model is characterized by the following three statements:

1. $Key_{CA} \Rightarrow CA$	Trust in the root key of the CA
2. CA controls $(Key \Rightarrow Principal)$	CA's Jurisdiction
3. Key_{CA} says $(Key \Rightarrow Principal)$	Certificate

Users trust that the root key belongs to the CA. Trust in the key of the root CA must be established by a trustworthy key distribution process. The CA has jurisdiction over statements associating a key with a particular principal and issues PKI certificates, each of which is a statement signed by the root key that associates the key with a particular principal. The PKI model does not deal with authorizations, and authorization is considered the responsibility of the relying party (RP). Moreover, validation is also seen as the responsibility of the RP, and does not involve the user.

The PKM model is characterized by the following two statements:

1. $Bank$ controls $(Key \Rightarrow Principal)$	Bank's Jurisdiction
2. $\langle Key, Validated\rangle \supset Bank$ says $(Key \Rightarrow Principal)$	Bank issues authorization

The PKM model blends authentication with authorization, and Banks have the authority for authenticating and authorizing the use of credentials. The user has the freedom to obtain credentials from any provider, but the Banks reinterpret the credentials in constrained manner, which could vary between banks. In contrast to the PKI model, the validation process for the credentials is explicit and involves the user, supporting non-repudiation claims. In effect, PKPS maps the common interpretation of PKI credentials into the more constrained and controlled interpretation required by banks for high-value commercial transactions.

6 Related Work

There are several XML schemas for specifying web service policies and privacy policies. WS-Policy [1] is a W3C standard for specifying web service policies for security, quality of service, messaging, etc. WSPL [4] has similar motivations, but is not an accepted W3C standard. P3P enables a web site to publish its privacy practice in a machine readable format, which all browsers can read and warn their respective users if the privacy practice of a web site is incompatible with a user's personal preference [5]. Our work relates to existing XML schemas for specifying web service and privacy policies by providing a formal semantics with sound inference rules for describing policies. The benefit of our work is banks can rigorously justify acting on payment instructions based on policies and trust assumptions.

Jon Olnes [6] describes an approach that offers interoperability by using a trusted third party called the validation authority (VA). The VA is trusted by both the CAs and relying party (RP), which receives the credentials. Each VA vouches for the CAs it handles, and the RP can validate all the credentials from the CAs by connecting to a single VA. While this model provides interoperability with respect to CAs vouched for by a particular VA, it limits the RP and its customers to only those CAs. In contrast, PKM imposes no such restrictions; Banks use any CAs they want. Moreover, the PKM model reinterprets the authority of credentials in a constrained and controlled manner.

Fox and LaMacchia [7] describe an alternative to OCSP for online certificate status checking. Any method similar to OCSP that requires the RP to connect to the CA for validating the certificates, not only breaks interoperability, but also imposes a significant cost on the RP. In contrast, PKM supports a general validation model, including a sender validation model, which in conjunction with the reinterpretation of authority, scales better, provides interoperability, and reduces the cost for the RP.

Our work is related to several logical systems used for reasoning about access-control that are summarized in [8]. The access-control logic we use is based on Abadi and Plotkin's work [9], with modifications described in [10].

7 Conclusion

The common interpretation of PKI credentials is problematic for banks engaged in high-value commercial transactions. Partner key management (PKM), through the use of partner key practice statements (PKPS), reinterprets PKI credentials to address the problems of scope of authority, liability, and cost inherent to high-value commercial transactions. The failure of any single high-value transaction can bring severe consequences to banks. Thus, it is essential that the policies and requirements regarding the use of credentials in high-value commercial transactions be as precise and accurate as possible. To meet this requirement, we have expressed PKI, PKPS, and PKM policies and interpretations in an access-control logic with formal semantics and sound inference rules. This enables banks and their customers to know precisely what is required of them and to justify acting on payment instructions. Our experience to date indicates that using this logic is within the capabilities of practitioners and does in fact clarify the underlying logic of credentials and their use in high-value commercial transactions.

References

1. Vedamuthu, A.S., Orchard, D., Hirsch, F., Hondo, M., Yendluri, P., Boubez, T.: Yalçinalp, Ü.: Web services policy 1.5 - framework (September 2007), `http://www.w3.org/TR/ws-policy/`
2. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. In: RFC 2560 (Proposed Standard) (June 1999)
3. Benson, G.: Portable security transaction protocol. Comput. Netw. 51(3), 751–766 (2007)
4. Anderson, A.H.: An introduction to the web services policy language (wspl). In: POLICY (2004)
5. Cranor, L., Dobbs, B., Egelman, S., Hogben, G., Humphrey, J., Langheinrich, M., Marchiori, M., Presler-Marshall, M., Reagle, J., Schunter, M., Stampley, D.A., Wenning, R.: The platform for privacy preferences 1.1 (p3p1.1) specification. (November 2006), `http://www.w3.org/TR/P3P11/`
6. Olnes, J.: DNV VA white paper: PKI interoperability by an independent, trusted validation authority. In: 5th Annual PKI R & D Workshop (April 2006)
7. Fox, B., LaMacchia, B.A.: Online certificate status checking in financial transactions: The case for re-issuance. In: FC (1999)
8. Abadi, M.: Logic in access control (tutorial notes), 145–165 (2009)
9. Abadi, M., Burrows, M., Lampson, B., Plotkin, G.: A Calculus for Access Control in Distributed Systems. ACM Transactions on Programming Languages and Systems 15(4), 706–734 (1993)
10. Chin, S.K., Older, S.: Reasoning about delegation and account access in retail payment systems. In: MMM-ACNS (2007)

A New Hard Problem over Non-commutative Finite Groups for Cryptographic Protocols

Dmitriy N. Moldovyan and Nikolay A. Moldovyan

St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences,
14 Liniya, 39, St. Petersburg 199178, Russia
mdn.spectr@mail.ru
http://www.spiiras.nw.ru

Abstract. A new computationally difficult problem defined over non-commutative finite groups is proposed as cryptographic primitive. The problem is used to construct public key agreement protocol and algorithms for public and commutative encryption. Finite non-commutative groups of the four-dimension vectors over the ground field are constructed and investigated as primitives for implementing the protocols and algorithms based on the proposed difficult problem.

Keywords: public key cryptography, difficult problem, finite non-commutative groups, public key distribution, public encryption, commutative encryption.

1 Introduction

Factorization and finding discrete logarithm are two of the most widely used in the public key cryptography difficult problems. The second problem is used in the official signature standards [1]. However both of this problems can be solved in polynomial time on a quantum computer [2]. Quantum computing develops from theoretic models towards practical implementations therefore cryptographers look for some new hard problems that have exponential complexity while using both the ordinary computers and the quantum ones [3,4]. Such new difficult problems have been defined over braid groups representing a particular type of infinite non-commutative groups. Using the braid groups as cryptographic primitive a number of new public key cryptosystems have been developed [5,6]. Unfortunately, results of the paper [7] show weakness of the conjugacy search problem used in the braid group based cryptographic protocols.

Present paper introduces a new hard problem defined over finite non-commutative groups and describes the public key cryptoschemes constructed using the proposed hard problem that combines the discrete logarithm problem with the conjugacy search problem. There is also presented a theorem disclosing the local structure of the non-commutative group, which is exploited in the proposed hard problem. Then concrete type of the non-commutative finite groups is constructed over finite four-dimension vector space.

2 New Hard Problem and Its Cryptographic Applications

Suppose for some given finite non-commutative group Γ containing element Q possessing large prime order q there exists a method for easy selection of the elements from

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 183–194, 2010.

sufficiently large commutative subgroup $\Gamma_{\rm ab} \in \Gamma$. One can select a private key as the pair (W,x) containing a random element $W \in \Gamma_{\rm ab}$ such that $W \circ Q \neq Q \circ W$, where $\circ$ denotes the group operation, and a random number $x < q$ and then compute the public key $Y = W \circ Q^x \circ W^{-1}$ (note that it is easy to show that for arbitrary value x the inequality $W \circ Q^x \neq Q^x \circ W$ holds). Finding pair (W,x), while given Γ, $\Gamma_{\rm ab}$, Q, and Y, is a computationally difficult problem that is suitable to design new public key cryptosystems. The problem suits also for designing commutative encryption algorithms. While constructing cryptoschemes on the basis of this hard problem there is used the mutual commutativity of the exponentiation operation and the automorphic mapping operation $\varphi_W(V) = W \circ V \circ W$, where V takes on values of all elements of the group Γ. The commutativity of these two operation can be expressed by the equality $\varphi_W(V^x) = (\varphi_W(V))^x$. Indeed, it is known [8] that

$$W \circ V^x \circ W^{-1} = \left(W \circ V \circ W^{-1}\right)^x.$$

The public key agreement protocols can be constructed as follows. Suppose two users have intension to generate a common secret key using a public channel. The first user generates his private key (W_1,x_1), computes his public key $Y_1 = W_1 \circ Q^{x_1} \circ W_1^{-1}$, and sends Y_1 to the second user. The last generates his private key (W_2,x_2), computes his public key $Y_2 = W_2 \circ Q^{x_2} \circ W_2^{-1}$, and sends Y_2 to the first user. Then, like in the Diffie-Hellman protocol [9], the first user computes the value

$$K_{12} = W_1 \circ (Y_2)^{x_1} \circ W_1^{-1} = W_1 \circ \left(W_2 \circ Q^{x_2} \circ W_2^{-1}\right)^{x_1} \circ W_1^{-1} = \\ = W_1 \circ W_2 \circ Q^{x_2 x_1} \circ W_2^{-1} \circ W_1^{-1}.$$

The second user computes the value

$$K_{21} = W_2 \circ (Y_1)^{x_2} \circ W_2^{-1} = W_2 \circ \left(W_1 \circ Q^{x_1} \circ W_1^{-1}\right)^{x_2} \circ W_2^{-1} = \\ = W_2 \circ W_1 \circ Q^{x_1 x_2} \circ W_1^{-1} \circ W_2^{-1}.$$

The elements W_1 and W_2 belong to the commutative subgroup $\Gamma_{\rm ab}$, therefore $K_{21} = K_{12} = K$, i.e. each of the users has generated the same secret K that can be used, for example, to encrypt confidential messages send through the public channel.

Suppose a public-key reference book is issued. Any person can send to some user a confidential message M using user's public key $Y = W \circ Q^x \circ W^{-1}$, where W and x are elements of user's private key. For this aim the following public key encryption scheme can be used, in which it is supposed using some encryption algorithm F_K controlled with secret key K representing an element of the group Γ.

1. Sender generates a random element $U \in \Gamma_{\rm ab}$ and a random number u, then computes the elements $R = U \circ Q^u \circ U^{-1}$ and
$$K = U \circ Y^u \circ U^{-1} = U \circ \left(W \circ Q^x \circ W^{-1}\right)^u \circ U^{-1} = U \circ W \circ Q^{xu} \circ W^{-1} \circ U^{-1}.$$

2. Using the element K as encryption key and encryption algorithm E_K sender encrypts the message M into the cryptogram $C = F_K(M)$. Then he sends the cryptogram C and element R to the user.

3. Using the element R the user computes the encryption key K as follows $K = W \circ R^x \circ W^{-1} = W \circ \left(U \circ Q^u \circ U^{-1}\right)^x \circ W^{-1} = W \circ U \circ Q^{ux} \circ U^{-1} \circ W^{-1}$. Then the user decrypts the cryptogram C as follows $M = F_K^{-1}(C)$, where F_K^{-1} is the decryption algorithm corresponding to the encryption algorithm F_K.

The proposed hard problem represents some combining the exponentiation procedure with the procedure defining the group mapping that is an automorphism. These two procedures are commutative therefore their combination can be used to define the following commutative-encryption algorithm.

1. Represent the message as element M of the group Γ.

2. Encrypt the message with the first encryption key (W_1, e_1), where $W_1 \in \Gamma_{\text{ab}}$, e_1 is a number invertible modulo m, and m is the least common multiple of all element orders in the group Γ, as follows $C_1 = W_1 \circ M^{e_1} \circ W_1^{-1}$.

3. Encrypt the cryptogram C_1 with the second encryption key (W_2, e_2), where $W_2 \in \Gamma_{\text{ab}}$, e_2 is a number invertible modulo m, as follows

$$C_{12} = W_2 \circ C_1^{e_2} \circ W_2^{-1} = W_2 \circ W_1 \circ M^{e_1 e_2} \circ W_1^{-1} \circ W_2^{-1}.$$

It is easy to show the encrypting the message M with the second key (W_2, e_2) and then with the first key (W_1, e_1) produces the cryptogram $C_{21} = C_{12}$, i.e. the last encryption procedure is commutative.

3 On Selection of the Elements from Commutative Subgroups

In the cryptoschemes described in previous section the first element of the private key should be selected from some commutative group. A suitable way to define such selection is the following one. Generate an element $G \in \Gamma$ having sufficiently large prime order g and define selection of the element W as selection of the random number $1 < w < g$ and computing $W = G^w$. Using this mechanism the private key is selected as two random numbers w and x and the public key is the element $Y = G^w \circ Q^x \circ G^{-w}$. One can easily show that for arbitrary values w and x the inequality $G^w \circ Q^x \neq Q^x \circ G^w$ holds.

For security estimations it represents interest haw many different elements are generated from two given elements G and Q having prime orders g and q, respectively. The following theorem gives a reasonable answer to this question.

Theorem 1. *Suppose elements G and Q of some non-commutative finite group Γ have the prime orders g and q, correspondingly, and satisfy the following expressions $G \circ Q \neq Q \circ G$ and $K \circ Q \neq Q \circ K$, where $K = G \circ Q \circ G^{-1}$. Then all of elements $K_{ij} = G^j \circ Q^i \circ G^{-j}$, where $i = 1, 2, \ldots, q-1$ and $j = 1, 2, \ldots, g$, are pairwise different.*

Proof. It is evident that for some fixed value j the elements $K_{ij} = G^j \circ Q^i \circ G^{-j}$, where $i = 1, 2, \ldots, q$, compose a cyclic subgroup of the order q. Condition $K \circ Q \neq Q \circ K$ means that element K is not included in the subgroup Γ_Q generated by different powers of Q. Suppose that for some values i, $i' \neq i$, j, and $j' \neq j$ elements K_{ij} and $K_{i'j'}$ are equal, i.e. $G^j \circ Q^i \circ G^{-j} = G^{j'} \circ Q^{i'} \circ G^{-j'}$. Multiplying the both parts of the last equation at the right by element G^j and at the left by element G^{-j} one gets $Q^i = G^{j'-j} \circ Q^{i'} \circ G^{-(j'-j)}$. The subgroup Γ_Q has the prime order, therefore its arbitrary element different from the unity element is generator of Γ_Q, i.e. for $i' \leq q-1$ the element $P = Q^{i'}$ generates subgroup Γ_Q. Taking this fact into account one can write

$$(Q^i)^z = \left(G^{j'-j} \circ Q^{i'} \circ G^{-(j'-j)}\right)^z = G^{j'-j} \circ Q^{i'z} \circ G^{-(j'-j)} = G^{j'-j} \circ P^z \circ G^{-(j'-j)} \in \Gamma_Q.$$

The last formula shows that mapping $\varphi_{G^{j'-j}}(P^z) = G^{j'-j} \circ P^z \circ G^{-(j'-j)}$ maps each element of Γ_Q in some element of Γ_Q. The mapping $\varphi_{G^{j'-j}}(\Gamma_Q)$ is bijection, since for $z = 1, 2, \ldots, q$ the set of elements $\left(Q^i\right)^z$ composes the subgroup Γ_Q. Thus, the mapping $\varphi_{G^{j'-j}}(\Gamma_Q)$ is a bijection of the subgroup Γ_Q into itself.

Since order of the element G is prime, there exists some number $u = (j' - j)^{-1} \bmod g$ for which the following expressions hold $G = \left(G^{j'-j}\right)^u$ and

$$\varphi_G(\Gamma_Q) = \varphi_{\left(G^{j'-j}\right)^u}(\Gamma_Q) = \underbrace{\varphi_{G^{j'-j}}\left(\varphi_{G^{j'-j}}\left(\ldots\varphi_{G^{j'-j}}(\Gamma_Q)\ldots\right)\right)}_{u \text{ bijections}},$$

where the mapping is represented as superposition of u mappings $\varphi_{G^{j'-j}}(\Gamma_Q)$. The superposition is also a bijection of the subgroup Γ_Q into itself, since the mapping $\varphi_{G^{j'-j}}(\Gamma_Q)$ is the bijection Γ_Q into Γ_Q. Therefore the following expressions hold:

$$K = G \circ Q \circ G^{-1} = \varphi_G(Q) \in \Gamma_Q \;\Rightarrow\; K \circ Q = Q \circ K.$$

The last formula contradicts to the condition $K \circ Q \neq Q \circ K$ of the theorem. This contradiction proves Theorem 1. □

Accordingly to Theorem 1 there exist $(q-1)g$ different elements $Z_{ij} \neq E$, where E is unity element of Γ. Together with the unity element E they compose g cyclic subgroups of the order q and each of elements $Z_{ij} \neq E$ belongs only to one of such subgroups.

4 Non-commutative Finite Rings of Four-Dimension Vectors

Different finite rings of m-dimension vectors over the ground field $GF(p)$, where p is a prime, can be defined using technique proposed in [10]. The non-commutative rings of four-dimension vectors are defined as follows. Suppose **e**, **i**, **j**, **k** be some formal basis vectors and $a, b, c, d \in GF(p)$, where $p \geq 3$, are coordinates. The vectors are denoted as $a\mathbf{e} + b\mathbf{i} + c\mathbf{i} + d\mathbf{k}$ or as (a, b, c, d). The terms $\tau\mathbf{v}$, where $\tau \in GF(p)$ and $\mathbf{v} \in \{\mathbf{e}, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$, are called components of the vector.

The addition of two vectors (a, b, c, d) and (x, y, z, v) is defined via addition of the coordinates corresponding to the same basis vector accordingly to the following formula

$$(a, b, c, d) + (x, y, z, v) = (a + x, b + y, c + z, d + v).$$

The multiplication of two vectors $a\mathbf{e} + b\mathbf{i} + c\mathbf{j} + z\mathbf{w}$ and $x\mathbf{e} + y\mathbf{i} + z\mathbf{j} + v\mathbf{k}$ is defined as multiplication of each component of the first vector with each component of the second vector in correspondence with the following formula

$$(a\mathbf{e} + b\mathbf{i} + c\mathbf{j} + z\mathbf{w}) \circ (x\mathbf{e} + y\mathbf{i} + z\mathbf{j} + v\mathbf{k}) = ax\mathbf{e} \circ \mathbf{e} + bx\mathbf{i} \circ \mathbf{e} + cx\mathbf{j} \circ \mathbf{e} + dx\mathbf{k} \circ \mathbf{e} +$$
$$+ az\mathbf{e} \circ \mathbf{j} + bz\mathbf{i} \circ \mathbf{j} + cz\mathbf{j} \circ \mathbf{j} + dz\mathbf{k} \circ \mathbf{j} + av\mathbf{e} \circ \mathbf{k} + bv\mathbf{i} \circ \mathbf{k} + cv\mathbf{j} \circ \mathbf{k} + dv\mathbf{k} \circ \mathbf{k},$$

where $\circ$ denotes the vector multiplication operation. In the final expression each product of two basis vectors is to be replaced by some basis vector or hy a vector containing only one non-zero coordinate in accordance with the basis-vector multiplication table

Table 1. The basis-vector multiplication table

$\circ$	$\vec{e}$	$\vec{\imath}$	$\vec{j}$	$\vec{k}$
$\vec{e}$	e	i	j	k
$\vec{\imath}$	i	$-\tau$e	k	$-\tau$j
$\vec{j}$	j	$-$k	$-$e	i
$\vec{k}$	k	τj	$-$i	$-\tau$e

(BVMT) defining associative and non-commutative multiplication. There are possible different types of the BVMTs, but in this paper there is used the BVMT of some particular type shown in Table 1. For arbitrary value $\tau \in GF(p)$ Table 1 defines formation of the non-commutative finite ring of four-dimension vectors. In the defined ring the vector $(1,0,0,0)$ plays the role of the unity element. For implementing the cryptoschemes described in Section 2 it represents interest to consider the multiplicative group Γ of the constructed non-commutative ring. To generate the elements Q and G of sufficiently large orders it is required computing the group order Ω that is equal to the number of invertible vectors. If some vector $A = (a,b,c,d)$ is invertible, then there exists its inverses $A^{-1} = (x,y,z,v)$ for which the following formula holds $A \circ A^{-1} = E = (1,0,0,0)$. This vector equation defines the following system of four linear equations with four unknowns x, y, z, and v:

$$\begin{cases} ax - \tau by - cz - \tau dv = 1 \\ bx + ay - dz + cv = 0 \\ cx + \tau dy + az - \tau bv = 0 \\ dx - cy + bz + av = 0. \end{cases} \tag{1}$$

If this system of equations has solution, then the vector (a,b,c,d) is invertible, otherwise it is not invertible. The main determinant of the system is the following one

$$\Delta(A) = \begin{vmatrix} a & -\tau b & -c & -\tau d \\ b & a & -d & c \\ c & \tau d & a & -\tau b \\ d & -c & b & a \end{vmatrix} \tag{2}$$

Computation of the determinant gives

$$\Delta(A) = \left(a^2 + \tau b^2 + c^2 + \tau d^2\right)^2. \tag{3}$$

Counting the number of different solutions of the congruence $\Delta(A) \equiv 0 \bmod p$ one can define the number N of non-invertible vectors and then define the group order $\Omega = p^4 - N$. The indicated congruence has the same solutions as the congruence

$$a^2 + \tau b^2 + c^2 + \tau d^2 \equiv 0 \bmod p. \tag{4}$$

Statement 1. *For prime $p = 4k + 1$, where $k \geq 1$ and $\tau \neq 0$, the order of the non-commutative group of the four-dimension vectors is equal to $\Omega = p(p-1)(p^2-1)$.*

Proof. For primes $p = 4k+1$ the number -1 is a quadratic residue, since $(-1)^{(p-1)/2} = (-1)^{2k} \equiv 1 \bmod p$. Therefore there exists number λ such that $\lambda^2 \equiv -1 \bmod p$ and congruence (4) can be represented as follows

$$\begin{gathered}
a^2 - (\lambda c)^2 \equiv \tau\left((\lambda b)^2 - d^2\right) \bmod p; \\
(a - \lambda c)(a + \lambda c) \equiv \tau\left((\lambda b)^2 - d^2\right) \bmod p; \\
\alpha\beta \equiv \tau\left((\lambda b)^2 - d^2\right) \bmod p,
\end{gathered}$$

where $\alpha \equiv a - \lambda c \bmod p$ and $\beta \equiv a + \lambda c \bmod p$. It is easy to see that for each pair of numbers (α, β) satisfying the last congruence correspond unique pair of numbers (a, c) satisfying congruence (4). Therefore the number of solutions of congruence (4) can be computed as number of solutions of the last equation. Two cases can be considered. The first case correspond to condition $(\lambda b)^2 - d^2 \not\equiv 0 \bmod p$ and there exist $(p-1)^2$ of different pairs (b, d) satisfying this condition. For each of such pairs (b, d) for all $(p-1)$ values $\alpha \not\equiv 0 \bmod p$ there exists exactly one value β such that the last congruence holds. Thus, the first case gives $N_1 = (p-1)^3$ different solutions of congruence (4).

The second case correspond to condition $(\lambda b)^2 - d^2 \equiv 0 \bmod p$ which is satisfied with $2p-1$ different pairs (b, d). The left part of the last congruence is equal to zero modulo p in the following subcases i) $\alpha \not\equiv 0 \bmod p$ and $\beta \equiv 0 \bmod p$ ($p-1$ different variants), ii) $\alpha \equiv 0 \bmod p$ and $\beta \not\equiv 0 \bmod p$ (there exist $p-1$ different variants), and iii) $\alpha \equiv 0 \bmod p$ and $\beta \equiv 0 \bmod p$ (one variant). Thus, the subcases gives $2p-1$ different variants of the pairs (a, c), therefore the second case gives $N_2 = (2p-1)^2$ different solutions of congruence (4). In total we have $N = N_1 + N_2 = (p-1)^3 + (2p-1)^2 = p^3 + p^2 - p$ solutions. The value N is equal to the number of non-invertible vectors and defines the group order $\Omega = p^4 - N = p^4 - p^3 - p^2 + p = p(p-1)(p^2-1)$. Statement 1 is proved. □

Statement 2. *Suppose prime $p = 4k+3$, where $k \geq 1$, $\tau \neq 0$, and the value τ is a quadratic non-residue modulo p. Then the order of the non-commutative group of four-dimension vectors is equal to $\Omega = p(p-1)(p^2-1)$.*

Proof. For primes $p = 4k+3$ the number -1 is a quadratic non-residue, since $(-1)^{(p-1)/2} = (-1)^{2k+1} \equiv -1 \bmod p$. Since the value τ is a quadratic non-residue the following formulas hold: $\tau^{(p-1)/2} \equiv -1 \bmod p$ and $(-\tau)^{(p-1)/2} \equiv 1 \bmod p$. The last formula shows that there exists number λ such that $\lambda^2 \equiv -\tau \bmod p$ and congruence (4) can be represented as follows

$$\begin{gathered}
a^2 - (\lambda b)^2 \equiv (\lambda d)^2 - c^2 \bmod p; \\
(a - \lambda b)(a + \lambda b) \equiv (\lambda d)^2 - c^2 \bmod p; \\
\gamma\delta \equiv (\lambda d)^2 - d^2 \bmod p,
\end{gathered}$$

where $\gamma \equiv a - \lambda b \bmod p$ and $\delta \equiv a + \lambda b \bmod p$. Then, counting different solutions of the last equation is analogous to counting solutions in the proof of Statement 1. This gives $N = p^3 + p^2 - p$ different solutions of congruence (4) and the group order $\Omega = p(p-1)(p^2-1)$. □

5 Homomorphism of the Vector Group

There exists a homomorphism of the group of four-dimension vectors Γ into the field $GF(p)$.

Theorem 2. *Suppose the vector A takes on all values of the elements of the group Γ. The determinant (2) defines the homomorphism $\psi(A) = \Delta(A)$ of the group Γ into the field $GF(p)$.*

Proof. Let us consider the vector equation

$$A \circ X = V \tag{5}$$

over the four-dimension vector space $\{V\}$, where A is an invertible vector and V is an arbitrary vector. Since $\Delta(A) \neq 0$ (see formula (2)), the equation (5) has unique solution for each vector V. Therefore multiplication of the vector A by all vectors $V \in \{V\}$defines a linear transformation T_A of $\{V\}$. The matrix M_A of coefficients of the system of equations (1) can be put into correspondence to T_A (see determinant of this matrix in formula (2)). Another invertible vector B defines the transformation T_B corresponding to analogous matrix M_B. The vector multiplication operation is associative, therefore we have

$$(A \circ B) \circ X = A \circ (B \circ X). \tag{6}$$

The left part of formula (6) represents the linear transformation $T_{A \circ B}$ corresponding to the matrix $M_{A \circ B}$. The right part of formula (6) is the superposition $T_B * T_A$ of linear transformations T_B and T_A, therefore we have

$$T_{A \circ B} = T_B * T_A \Rightarrow M_{A \circ B} = M_A M_B \Rightarrow$$
$$\Rightarrow \Delta(A \circ B) = \Delta(A)\Delta(B).$$

The last expression means that the mapping $\psi : A \to \Delta(A)$ is the homomorphism of the group Γ into the field $GF(p)$. Theorem 2 is proved. □

Using different BVMT defining associative multiplication of the m-dimension vectors defined over the finite fields $GF(p^s)$, where $s \geq 1$, one can define different finite vector groups, commutative [10] and non-commutative. Theorem 1 can be easily extended to all of such vector groups, i.e. the determinant $\Delta(A)$ of the system of equations providing computation of the inverses of the vector A defines the homomorphism of any of such groups into the field $GF(p^s)$.

This homomorphism should be taken into account while selecting the parameters of the public key agreement protocol and of the public encryption algorithm based on the proposed hard problem. Indeed, in the case of using the group Γ the vector Q should have the order q such that $q | p+1$ and $q \not| p-1$. In this case the homomorphism maps the public key into the unity element of the field $GF(p)$. This is stated by the following statement.

Statement 3. *If the vector V has the order ω_V such that* $\gcd(\omega_V, p-1) = 1$, *then* $\Delta(V) = 1$.

Proof. Suppose $\Delta(V) \neq 1$. Then we have

$$\{\Delta(V^{\omega_V}) = \Delta(E) = 1 \text{ and } \Delta(V^{\omega_V}) = (\Delta(V))^{\omega_V}\} \Rightarrow (\Delta(V))^{\omega_V} = 1 \Rightarrow$$
$$\Rightarrow \gcd(\omega_V, p-1) \neq 1.$$

The last expression contradicts to the condition $\gcd(\omega_V, p-1) = 1$ of the statement. This contradiction proves Statement 3. □

In the case of incorrect selection of the vector Q the secrete key (x, W) can be computed by parts solving two independent hard problems, the discrete logarithm problem and the conjugacy search problem. For example, suppose the vector Q has the order q such that $q|p-1$. Then we have

$$\Delta(Y) = \Delta(W)(\Delta(Q))^x(\Delta(W))^{-1} = (\Delta(Q))^x,$$

where $\Delta(Q) \neq 1$, and the value x can be found solving the discrete logarithm problem in $GF(p)$. Then the value W can be found solving the conjugacy search problem defined by equation $Y = W \circ V \circ W^{-1}$, where Y and $V = Q^x$ are known vectors. The discrete logarithm can be found in polynomial time using the known algorithm for quantum computations proposed by P. Shor [2]. Therefore using the quantum computer the proposed problem can be reduced in polynomial time to the conjugacy search problem, if $q|p-1$.

In the case of large prime order $\omega(Q) = q$ such that $q|p+1$ and $q \not| p-1$ this attack does not work. Since the conjugacy search problem is considered as a primitive for post quantum cryptography and the proposed problem in the case $q|p+1$ is harder than both the dicrete logarithm and the conjugacy search problem we suppose the proposed cryptoschemes effectively resist the quantum attacks.

6 Complexity of the Private-Key Computation in a Particular Case

Using the known parameters Q and G having the orders q and $g = q$ the following algorithm finds the private key (w,x) from the public one $Y = G^w \circ Q^x \circ G^{-w}$.

1. For all values $j = 1, 2, \ldots, q$ compute vectors $U(j) = G^j \circ Y \circ G^{-j}$ (difficulty of this step is $2q$ vector multiplications).
2. Order the table computed at the step 1 accordingly to the values $U(j)$ (difficulty of this step is $q\log_2 q$ comparison operations).
3. Set counter $i = 1$ and initial value of the vector $V = (1, 0, 0, 0)$.
4. Compute the vector $V \leftarrow V \circ Q$.
5. Check if the value V is equal to some of the vectors $U(j)$ in the ordered table. If there is some vector $U(j') = V$, then deliver the private key $(w,x) = (j', i)$ and STOP. Otherwise go to step 6.
6. If $i \neq q$, then increment counter $i \leftarrow i + 1$ and go to step 4. Otherwise STOP and output the message INCORRECT CONDITION. (Difficulty of steps 5 and 6 does not exceed q vector multiplication operations and $q\log_2 q$ comparison operations.)

Overall the time complexity of this algorithm is about $3q$ vector multiplication operations and $2q\log_2 q$ comparison operations, i.e. the time complexity is $O(q)$ operations, where $O(\cdot)$ is the order notation. The algorithm requires storage for q vectors and for the same number of $|p|$-bit numbers, i.e. the space complexity is $O(q)$.

This algorithm shows that the 80-bit security of the proposed cryptosystems can be provided selecting 80-bit primes q and g. Such prime orders of the vectors Q and G can be get using 81-bit primes p.

It seems that element G having composite order can be used in the cryptoschemes described above and this will give higher security, while using the given fixed modulus p. However this item represents interest for independent research.

7 Experiments and Numerical Illustrations

Numerous computational experiments have shown that in the case $p = 4k + 3$, where $k \geq 1$ and $\tau \neq 0$, when the value τ is a quadratic residue modulo p, the group order also equals to $\Omega = p(p-1)(p^2-1)$. However the formal proof of the last fact have not been found. The experiments have also shown that for given modulus p the structure of the non-commutative group of four-dimension vectors is the same for all non-zero values of the structural coefficient τ. Here under structure of the group it is supposed a table showing the number of different vectors having the same order ω for all possible values ω. In the case of the commutative finite groups of four-dimension vectors the group structure changes with changing values of structural coefficients. The experiments have been performed using different other variants (than Table 1) of the BVMTs defining non-commutative groups of four-dimension vectors and in all cases the same structure and the same group order have been get, for all non-zero values of the structural coefficients.

Defining a group of four-dimension vectors with Table 1 and parameters $\tau = 1$ and $p = 234770281182692326489897$ (it is a 82-bit number) one can easily generate the vectors Q and G having the prime orders $q = g = 117385140591346163244949$ (it is a 81-bit number; $q = (p+1)/2$) and then generate vector $K = G \circ Q \circ G^{-1}$:

$$\begin{aligned}
Q &= (197721689364623475468796, 104620049500285101666611,\\
&\quad 91340663452028702293061, 190338950319800446198610);\\
G &= (44090605376274898528561, 33539251770968357905908,\\
&\quad 62849418993954316199414, 121931076128999477030014);\\
G^{-1} &= (44090605376274898528561, 201231029411723968583989,\\
&\quad 171920862188738010290483, 112839205053692849459883);\\
K &= (197721689364623475468796, 127324294038715727080605,\\
&\quad 205837389432865711027118, 169402831102520905889980).
\end{aligned}$$

The vectors satisfy the conditions $G \circ Q \neq G \circ Q$ and $K \circ Q \neq Q \circ K$ (see Theorem 1), therefore they can be used to implement the cryptoschemes presented in Sections 2 and 3. It is easy to generate many other different pairs of the vectors Q and G possessing 81-bit prime orders q and g and satisfying the condition of Theorem 1. The least common multiple of all element orders in the constructed group is

$$\begin{aligned}
m = {} & 129398535261883131443362128353893964593169206\,0\\
& 964758959029747196964737 6\,.
\end{aligned}$$

The exponent e of the encryption key for commutative encryption algorithm can be selected as $e = 736475851953646171911 7$. Then the exponent of the decryption key is computed using formula $d = e^{-1} \bmod m$:

$$d = 8969427630416482351904498868955232431090386202\\ 18896738106440367092666 1\ .$$

Accordingly to the algorithm for computing the private key from the public one, which is described in Section 6, the 80-bit security of the proposed cryptoschemes is provided in the case of 80-bit primes q and g. In this case the difficulty of the computation of the public key from the private one does not exceed 4000 multiplications modulo 81-bit prime. In the corresponding cryptoschemes of the public encryption and of the public key agreement, which are based on elliptic curves, the difficulty of computing the public key from the private one is equal to about 2400 multiplications modulo 160 prime. Taking into account that difficulty of the modulo multiplication is proportional to squared length of the modulus one can estimate that the proposed cryptoschemes are about 2.4 times faster than analogous schemes implemented using elliptic curves. Besides, performance of the proposed cryptoschems can be significantly enhanced defining computation of the secrete element W as a sum of small powers of G, for example, $W = \sum_{s=1}^{6} \rho_s G^{t_s}$, where $\rho_s \in GF(p)$, $t_s \leq 15$, $s = 1, 2, \ldots, 6$.

Experiments have shown that four each pair of vectors G and Q such that $G \circ Q \neq Q \circ G$ the condition $K \circ Q \neq Q \circ K$, where $K = G \circ Q \circ G^{-1}$, holds. One can suppose that the condition of Theorem 1 is excessive, however attempts to prove formally this theorem without condition $G \circ Q \neq Q \circ G$ were not successful. Probably there exist non-commutative groups for which condition $G \circ Q \neq Q \circ G$ does not lead to condition $K \circ Q \neq Q \circ K$. This is an item of our future research. As regards to selection of the elements G and Q that are to be used in the public key agreement protocol based on the considered hard problem one can check that for the selected elements G and Q all conditions of Theorems 1 and 2 are satisfied.

8 Finite Matrices Groups

For given value n all non-degenerate $n \times n$ matrices defined over the ground field $GF(p)$ compose a finite non-commutative group [8] having the order

$$\Omega_{n \times n} = \prod_{i=0}^{i=n-1} p^i (p^{n-i} - 1).$$

It is interesting that the order of the 2×2-matrix group is equal to the order of the four-dimension vector groups (in the case $\tau \neq 0$) described in Section 4: $\Omega_{2\times 2} = p(p-1)(p^2-1)$. (For the four-dimension vector groups defined using structural coefficient $\tau = 0$ the order is equal to $\Omega = p^2(p-1)^2$ for prime $p = 4k+1$ and to $\Omega = p^2(p^2-1)$ for prime $p = 4k+3$.)

In the cryptoschemes based on the proposed hard problem there are used the group elements having sufficiently large prime orders q and g that divide the group order. In the case of prime values n one can select the value p such that the value $q_{\max}$

$= \left(p^{n-1}\right) n^{-1}(p-1)^{-1}$ is prime and it is easy to generate matrices having the order $q_{\max}$. Taking this fact into account together with the fact that the matrix multiplication can be performed with n^3 arithmetic multiplications, about n^3 additions, and n^2 arithmetic divisions it is easy to come to conclusion that for practical applications the finite groups of matrices corresponding to the values $n = 2, 3, 5$, and 7 are of the most practical interest.

In case of the 3×3 matrices one can select such 42-bit prime p that the largest prime divisor of the group order $\Omega_{3\times3}$ is equal to 80-bit prime $q = \left(p^2 + p + 1\right)/3$ providing the 80-bit security of the proposed cryptoschemes with 378-bit public key. In the case of using the four-dimension vector groups or the 2×2 matrix group we get the same security with 324-bit public key.

For abitrary prime n one can find such primes p (for cases of different size of the value p) that value

$$q = \frac{p^{n-1} + p^{n-2} + \ldots + p + 1}{n}$$

is also prime. Since such value q divides $\Omega_{n\times n}$ one can use the values p having smaller size and get faster cryptoschems for the cases $n = 5$ and $n = 7$, however in the last two cases we get sufficiently large public keys (about 550 and 735 bits, respectively). A rough comparison of the time required for computing the common secret key using the Diffie-Hellman protocol based on different hard problems (see Table 2) shows that for the same security level the proposed hard problem provides faster key generation.

Table 2. Rough estimation of the time required for generating the common secret key with the Diffie-Hellman protocol implemented using different hard problems (in all cases the selected parameters provide the 80-bit security of the protocol)

Hard problem	Finite group	Size of prime p, bits	Time, arb. un.
Diserte logarithm	Elliptic curve over $GF(p)$	160	2200
Diserte logarithm	$\mathbb{F}_p^*$	1024	10000
Proposed	4-dimension vectors over $GF(p)$	81	350
Proposed	2×2 matrices over $GF(p)$	81	350
Proposed	3×3 matrices over $GF(p)$	42	200
Proposed	5×6 matrices over $GF(p)$	22	150
Proposed	7×7 matrices over $GF(p)$	15	150

9 Conclusion

Results of this paper shows that finite non-commutative groups represent interest for designing fast public key agreement schemes, public encryption algorithms, and commutative encryption algorithms. Such cryptoschemes are fast and the hard problem they are based on is expected to have exponential difficulty using both the ordinary computers and the quantum ones.

Theorems 1 and 2 are useful for justification of the selection elements Q and G while defining parameters of the cryptoschemes. The proposed non-commutative finite group of the four-dimension vectors seems to be appropriate for practical implementation of the proposed schemes. We have proved the formulas for computing the order of such groups in majority of cases. Unfortunately for a quarter of cases the formal proof have not been found and this item remains open for future consideration. However the proved cases coves the practical demands while implementing the proposed cryptoscheme with use of the composed non-commutative groups.

Implementation of the proposed cryptoschemes using the finite groups of matrices having size 3×3, 5×5, and 7×7 yields faster key generation, however in this case the size of public key is sufficiently large (from 378 to 735 bits). For designing fast cryptoschemes with sufficiently small public keys (320-330 bits) the finite non-commutative groups of the m-dimension vectors, where $m = 8, 16, 20, 28$, and 32, are very attractive. Construction and investigation of such finite groups of vectors represents a topic of independent research.

References

1. International Standard ISO/IEC 14888-3:2006(E). Information technology – Security techniques – Digital Signatures with appendix – Part 3: Discrete logarithm based mechanisms
2. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on quantum computer. SIAM Journal of Computing 26, 1484–1509 (1997)
3. Anshel, I., Anshel, M., Goldfeld, D.: An Algebraic Method for Public Key Cryptography. Mathematical Research Letters 6, 287–291 (1999)
4. Ko, K.H., Lee, S.J., Cheon, J.H., Han, J.W., Kang, J.S., Park, C.: New Public-Key Cryptosystems Using Braid Groups. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 166–183. Springer, Heidelberg (2000)
5. Lee, E., Park, J.H.: Cryptanalysis of the Public Key Encryption Based on Braid Groups. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 477–489. Springer, Heidelberg (2003)
6. Verma, G.K.: A Proxy Blind Signature Scheme over Braid Groups. International Journal of Network Security 9, 214–217 (2009)
7. Myasnikov, A., Shpilrain, V., Ushakov, A.: A Practical Attack on a Braid Group Based Cryptographic Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 86–96. Springer, Heidelberg (2005)
8. Kargapolov, M.I., Merzlyakov, Y. I.: Group Theory Foundations. Fizmatlit, Moscow (1996) (in Russian)
9. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Transactions on Information Theory IT-22, 644–654 (1976)
10. Moldovyan, N.A., Moldovyanu, P.A.: New Primitives for Digital Signature Algorithms: Vector Finite Fields. Quasigroups and Related Systems 17, 271–282 (2009)

Credential Chain Discovery in RT^T Trust Management Language

Krzysztof Sacha

Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warszawa, Poland
k.sacha@ia.pw.edu.pl

Abstract. The goal of this paper is to explore the potential of Role based Trust management language RTT as a means for specifying security policies and using credentials to ensure that confidential resources are not being granted to unauthorized users. The paper describes formally the syntax and semantics of the language and defines RTT credential graphs and credential chains as a means for answering security queries. Backward and forward search algorithms to build a credential chain are given.

Keywords: Software security, trust management, role-based trust management language, credential graph, credential chain.

1 Introduction and Related Work

Software systems, which are used in commercial, governmental and industrial sectors, store data and offer services that can be used safely by only a limited set of authorized users. Unauthorized access to data and other resources of such a system may have disastrous results. Therefore, construction of the mechanisms for controlling access to resident information and other resources of computer systems is one of the most important problems that must be solved by the information technology.

The traditional approach to access control relies on knowing the identity of all the entities that can make requests, and making decisions on allowing or denying access to system resources based on a verification of the identity of the requester. When the system grows and the number of entities becomes very big, they are divided into roles, i.e. overlapping groups of entities, which have the same rights and privileges with respect to the system resources [15,8]. This simplifies administration, however, the system must still know the members of each role and the access control is still based on a verification of identity. One possible mechanism of such a verification is a login window. Another example can be the use of a public key.

A much bigger problem arises is distributed open systems, in which the identity of users is not known in advance and can change in time outside the control of an access mediator. If this is the case, a new approach to access control is needed. For example, consider a scientific conference, which offers a reduction of the conference fee for members of the sponsor organizations. When I come to the registration desk and say that I am Chris Sacha, then my identity itself will not help in deciding whether I am eligible for a reduced fee or not. What can help, are two credentials stating that I am

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 195–208, 2010.

employed at an organization, and that the organization is a conference sponsor. Credentials can be implemented in a software system as digitally signed documents.

This paper deals with Role-based Trust management (RT) languages for describing security policies, roles and credentials in decentralized and open environments [1-6]. Credentials are statements in a RT language, describing entities (role issuers and requesters) and roles, which the entities can play in the system. The key concept of the trust management approach is delegation: An entity may transfer limited authority over a resource to other entities. Such a delegation can be implemented by means of an appropriate credential. This way, a set of credentials defines the security policy and allows of deciding on who is authorized to access a resource, and who is not.

The first trust management systems were PolicyMaker [1,2], KeyNote [3] and SPKI/SDSI [5]. All those systems used languages that allowed assigning privileges to entities and used credentials to delegate permissions from its issuer to its subject. A missing feature was the possibility of delegation based on attributes of the entities.

Role-based Trust management languages use roles to represent attributes [11]: A role is a set of entities who have the attribute represented by the role. There are several RT languages, with varying expressive power and complexity. The basic language RT_0 [13] allows describing roles, role hierarchies, delegation of authority over roles and role intersections. RT^T provides manifold roles to express threshold and separation of duties policies. A manifold role is a role that can be satisfied by a set of cooperating entities. A threshold policy requires a specified minimum number of entities to agree before access is granted. Separation of duties policy requires a set of entities, each of which fulfils a specific role, to agree before access is granted. Both types of policies mean that some roles cannot be fulfilled by a single entity and a set of entities must cooperate in order to satisfy these roles.

RT languages have well defined syntax [11,12] and intuitive meaning. A set-theoretic semantics has been defined for RT_0 in [13,9] and for RT^T in [7].

The rest of this paper is organized as follows. BNF syntax and an improved and simplified definition of the semantics of RT^T are described in Section 2. A credential graph and a credential chain, which allow answering the access control queries in RT^T, are presented in Section 3. Final remarks and plans for further research are described in Conclusions.

2 The Language RT^T

There are three basic elements in all the RT languages: Entities, role names and roles. *Entities* are actors within an access control system, which can participate in issuing permissions and making requests to access resources. An entity can, e.g., be a person or a program identified by a user account or a public key in a computer system. *Role names* represent permissions that can be granted to sets of entities (may be singleton sets) to manipulate resources. *Roles* represent sets of entities that have particular permissions granted according to the access control policy. The statements in RT^T are *credentials*, which are used for describing access control policies, assigning entities to roles and delegating authority to the members of other roles.

2.1 The Syntax

In this paper, we use nouns beginning with a capital letter or just capital letters, e.g. *A*, *B*, *C*, to denote sets of entities. Role names are denoted as identifiers beginning with a small letter or just small letters, e.g. *r*, *s*, *t*. Roles take the form of a set of entities (the issuer of this role) followed by a role name separated by a dot, e.g. $A.r$. A credential consists of a role, left arrow symbol and a valid role expression, e.g. $A.r \leftarrow e$.

BNF specification of the RT^T syntax can be written as follows.

```
<credential>        ::= <role> ← <role-expression>
<role>              ::= <entity-set> . <role-name>
<role-expression> ::= <entity-set>
                        | <role>
                        | <role> . <role-name>
                        | <role> ∩ <role
                        | <role> ⊕ <role>
                        | <role> ⊗ <role>
```

There are six types of role expressions and six types of credentials in RT^T, which are interpreted in the following way:

$A.r \leftarrow B$ – *simple membership*: a set of entities *B* can satisfy role $A.r$.

$A.r \leftarrow B.s$ – *simple inclusion*: role $A.r$ includes all members of role $B.s$. This is a delegation of authority over *r* from *A* to *B*, as *B* may cause new sets of entities to become members of the role $A.r$ by issuing credentials that define $B.s$.

A.r ← B.s.t – *linking inclusion*: role $A.r$ includes role $C.t$ for each *C*, which is a member of role $B.s$. This is a delegation of authority over *r* from *A* to all the members of the role $B.s$.

$A.r \leftarrow B.s \cap C.t$ – *intersection inclusion*: role $A.r$ includes all the sets of entities who are members of both roles $B.s$ and $C.t$. This is a partial delegation from *A* to *B* and *C*.

$A.r \leftarrow B.s \oplus C.t$ – role $A.r$ can be satisfied by a union set of one member of role $B.s$ and one member of role $C.t$. This allows expressing separation of duties policies.

$A.r \leftarrow B.s \otimes C.t$ – role $A.r$ can be satisfied by a union set of one member of role $B.s$ and one member of role $C.t$, where both members are disjoint sets of entities. This allows expressing threshold policies.

2.2 The Semantics

The syntax of a language describes the rules for constructing language expressions, such as credentials in RT^T. The semantics of a language describes the meaning of expressions in the application domain. A definition of semantics consists of two parts [10]: A semantic domain, which gives meaning to the language expressions, and a semantic mapping from the syntax to the semantic domain.

The semantics of RT^T defines the meaning of a set of credentials as a relation over a set of roles and the power set of entities. Thus, we use a Cartesian product of the set of roles and the power set of entities as the semantic domain of RT^T. The semantic mapping assigns a relation between roles and sets of entities to a set of credentials.

Let E be a set of entities and R be a set of role names. P is a set of RT^T credentials. The semantic domain of RT^T is a Cartesian product of sets:

$$2^E \times R \times 2^E$$

An instance of this product, e.g. (A, r, B) consists of a set A of entities that issue a role, the role name r and a set B of entities that fulfill the role $A.r$. If the cardinality of set B in (A, r, B) is greater than one, then the role $A.r$ is a manifold role.

The semantics of P, denoted by S_P, is a relation:

$$S_P \subseteq 2^E \times R \times 2^E$$

Let A, B, C, X, Y be arbitrary sets of entities (may be singletons) and r, s, t arbitrary role names. The semantics of RT^T can formally be defined in the following way.

Definition 1 (Semantics of RT^T). The semantics of a set P of RT^T credentials is the smallest relation $S_P \subseteq 2^E \times R \times 2^E$, which is closed with respect to the following properties:

- $(A, r, X) \in S_P$ for each $A.r \leftarrow X \in P$ (1)
- If $A.r \leftarrow B.s \in P$ and $(B, s, X) \in S_P$, then $(A, r, X) \in S_P$ (2)
- If $A.r \leftarrow B.s.t \in P$ and $(B, s, C) \in S_P$ and $(C, t, X) \in S_P$, then $(A, r, X) \in S_P$ (3)
- If $A.r \leftarrow B.s \cap C.t \in P$ and $(B, s, X) \in S_P$, $(C, t, X) \in S_P$, then $(A, r, X) \in S_P$ (4)
- If $A.r \leftarrow B.s \oplus C.t \in P$ and $(B, s, X) \in S_P$, $(C, t, Y) \in S_P$, then $(A, r, X \cup Y) \in S_P$ (5)
- If $A.r \leftarrow B.s \otimes C.t \in P$ and $(B, s, X) \in S_P$, $(C, t, Y) \in S_P$ and $X \cap Y = \phi$, then $(A, r, X \cup Y) \in S_P$ (6)

□

Definition 1 is recursive in that it defines new elements of S_P in relation to another elements of S_P. Resolving the recursion, we can construct S_P in a sequence of m steps, $m \geq 1$, which results in a sequence of m sets $S_0 \dots S_k \dots S_m$, such that:

1. $S_0 = \phi$
2. S_k, for $k \geq 1$, contains S_{k-1} and a triple (A, r, X) that has been derived from S_{k-1} by an application of one of the properties (1) through (6) in Definition 1.
3. If $S_{m+1} = S_m$, then $S_m = S_P$

Please note that $S_k \subseteq S_P$ for each $k \geq 0$. The algorithm is finite, i.e. the number of steps m is finite, because the power set of entities 2^E and the set of role names R are finite.

2.3 An Example

A company C has departments $D1$ and $D2$. There are company managers and there are accountants employed at each department. Such a structure of the company and the roles of employees can be described using *simple membership* credentials:

$$\{C\}.department \leftarrow \{D1\} \tag{7}$$
$$\{C\}.department \leftarrow \{D2\} \tag{8}$$
$$\{C\}.manager \leftarrow \{Adam\} \tag{9}$$
$$\{C\}.manager \leftarrow \{Bob\} \tag{10}$$
$$\{D1\}.accountant \leftarrow \{Adam\} \tag{11}$$
$$\{D1\}.accountant \leftarrow \{Alice\} \tag{12}$$
$$\{D2\}.accountant \leftarrow \{Betty\} \tag{13}$$

The accountants at the departments have the rights of company accountants. This is a delegation of role company accountant, issued by the company, to the members of role department accountant. This can be described using *simple inclusion* credentials:

$$\{C\}.accountant \leftarrow \{D1\}.accountant \tag{14}$$
$$\{C\}.accountant \leftarrow \{D2\}.accountant \tag{15}$$

or a single *linking inclusion* credential:

$$\{C\}.accountant \leftarrow \{C\}.department.accountant \tag{16}$$

A bank, which supports the company, requires that a company accountant approves a small transaction. A single person, who has the rights of a company accountant as well as of a manager, can approve a medium scale transaction. Such a policy of the bank can be described using *simple inclusion* and *intersection inclusion* credentials:

$$\{Bank\}.approveSmall \leftarrow \{C\}.accountant \tag{17}$$
$$\{Bank\}.approveMedium \leftarrow \{C\}.accountant \cap \{C\}.manager \tag{18}$$

Two accountants and a manager can jointly approve a big transaction. A manager who has the rights of an accountant can serve both roles at the transaction. Such a policy can be described using credentials:

$$\{C\}.twoAccountants \leftarrow \{C\}.accountant \otimes \{C\}.accountant \tag{19}$$
$$\{Bank\}.approveBig \leftarrow \{C\}.twoAccountants \oplus \{C\}.manager \tag{20}$$

The semantics of the above set of credentials can be constructed in several steps, according to recursive Definition 1. The construction is shown in Table 1.

Table 1. Construction of the semantics of RT^T credentials

Step	Semantics
1	$(\{C\}, department, \{D1\}), (\{C\}, department, \{D2\}), (\{C\}, manager, \{Adam\}),$ $(\{D1\}, accountant, \{Adam\}), (\{D1\}, accountant, \{Alice\}), (\{D2\}, accountant, \{Betty\}),$
2	$(\{C\}, accountant, \{Adam\}), (\{C\}, accountant, \{Alice\}), (\{C\}, accountant, \{Betty\}).$
3	$(\{Bank\}, approveSmall, \{Adam\}), (\{Bank\}, approveSmall, \{Alice\}),$ $(\{Bank\}, approveSmall, \{Betty\}).$
4	$(\{Bank\}, approveMedium, \{Adam\}),$
5	$(\{C\}, twoAccountants, \{Adam, Alice\}), (\{C\}, twoAccountants, \{Adam, Betty\}),$ $(\{C\}, twoAccountants, \{Alice, Betty\}).$
6	$(\{Bank\}, approveBig, \{Adam, Alice\}), (\{Bank\}, approveBig, \{Adam, Betty\}),$ $(\{Bank\}, approveBig, \{Alice, Betty, Adam\})$

3 Credential Chain

The rights to access resources are granted to roles, such as $A.r$ or $B.s$, which members are groups of entities (manifold roles). When a group X of entities submits a request to access a resource, then the access mediator needs to decide whether X is a member of the role, say $A.r$, which was granted access to this resource. One way to make such a decision could be to compute the semantics of the entire set of credentials that define the security policy and to check whether X belongs to role $A.r$ or not. Unfortunately, such an approach could be inefficient if the set of credentials was very large. A much better approach is to take into account not all the existing credentials, but only those that are necessary to decide on membership of X (the requester) in $A.r$ (the role authorized to access the resource).

3.1 Credential Graph

A credential graph, introduced for RT_0 in [13], is a graphical representation of the semantics of a set P of credentials. The nodes of the graph are role expressions, which appear within the credentials, and the directed edges reflect inclusion of sets that are the meaning of those expressions. Making a decision on the membership of an entity B in the role $A.r$ is equivalent to checking whether a path from B to $A.r$ exists in the graph or not. RT_0 credential graph is static in that the set E of entities that can issue roles, delegate permissions to other entities and make requests to access resources is constant. No new entity can be created by any credential of set P.

RT^T credential graph, introduced in Definition 2 below, is dynamic. Role issuers as well as requesters are groups of entities, and credentials of type $A.r \leftarrow B.s \oplus C.t$ and $A.r \leftarrow B.s \otimes C.t$ can create new groups of entities that can issue roles, delegate permissions to other groups of entities or make requests to access resources. Such a dynamic nature makes the construction of RT^T credential graph much more difficult.

Let P be a set of RT^T credentials over a set E of entities and a set R of role names.

Definition 2 (RT^T Credential Graph). RT^T credential graph is an ordered pair $G_P = (N_P, E_P)$ comprising a set N_P of nodes, which are role expressions that appear in credentials from P and subsets of entities from E, and a set E_P of directed edges, which are ordered pairs of nodes from N_P. The sets N_P and E_P are the smallest sets that are closed with respect to the following properties:

1. If a credential $A.r \leftarrow e$, where e is a role expression, belongs to P, then the nodes $A.r$ and e belong to N_P and a *credential edge* $(e, A.r)$ belongs to E_P.
2. If role expressions $B.s.t$ and $C.t$ belong to N_P and there exists a path from C to $B.s$ in G_P, then a *derived edge* $(C.t, B.s.t)$ belongs to E_P. The path from C to $B.s$ creates a *support set* for this edge.
3. If role expressions $B.s \cap C.t$ and X belong to N_P and there exist paths from X to $B.s$ and from X to $C.t$ in G_P, then a *derived edge* $(X, B.s \cap C.t)$ belongs to E_P. The paths from X to $B.s$ and from X to $C.t$ create a *support set* for this edge.
4. If role expressions $B.s \oplus C.t$, $B.s$, $C.t$, X, Y belong to N_P and there exit paths from X to $B.s$ and from Y to $C.t$ in G_P, then a *derived node* $X \cup Y$ belongs to N_P and a

derived edge ($X \cup Y$, $B.s \oplus C.t$) belongs to E_P. The paths from X to $B.s$ and from Y to $C.t$ create a *support set* for both derived elements.

5. If role expressions $B.s \otimes C.t$, $B.s$, $C.t$, X, Y belong to N_P and there exit paths from X to $B.s$ and from Y to $C.t$ in G_P, and $X \cap Y = \phi$, then a *derived node* $X \cup Y$ belongs to N_P and a *derived edge* ($X \cup Y$, $B.s \otimes C.t$) belongs to E_P. The paths from X to $B.s$ and from Y to $C.t$ create a support set for both derived elements.

□

Definition 2 is recursive in that it defines new elements of G_P in relation to another elements of G_P. Resolving the recursion, we can construct G_P in a sequence of m steps, $m \geq 1$, which result in a sequence of m subgraphs $G_1 \ldots G_k \ldots G_m$, such that:

1. G_1 is composed of credential nodes and credential edges, created by an application of property 1 in Definition 2 to all the credentials in P.
2. G_k is composed of G_{k-1} and a derived edge and (possibly) a derived node added by an application of one of the properties 2 through 5 in Definition 2 to G_{k-1}.
3. If $G_{m+1} = G_m$, then $G_m = G_P$.

Please note that $G_k \subseteq G_P$ for $k \geq 1$. The algorithm is finite, i.e. the number of steps m is finite, because the power set of entities 2^E and the set of role names R are finite.

3.2 Soundness and Completeness

Denote the power set of entities by $F = 2^E$. Each element in F is a set of entities from E. Each element in 2^F is a set, composed of sets of entities from E. The semantics of P can now be described as a function:

$$\hat{S}_P : 2^E \times R \rightarrow 2^F$$

that maps each role from $2^E \times R$ to a set of all such sets of entities, which are members of this role. Knowing the relation S_P, one can define the function $\hat{S}_P$ as follows:

$$\hat{S}_P(\, A.r\,) = \{\, X \in 2^E : (\, A, r, X\,) \in S_P \,\}$$

Let EX_P be the set of role expressions that appear within the credentials of set P. The function $\hat{S}_P$ can be extended to the domain of role expressions EX_P:

$$\hat{S}_P : EX_P \rightarrow 2^F$$

by adding the following six definitions, related to six types of RT^T credentials ($X \subseteq E$ is a set of entities, may be a singleton):

$$\hat{S}_P(\, X\,) = \{\, X \,\} \tag{21}$$

$$\hat{S}_P(\, A.r\,) = \{\, X \in 2^E : (\, A, r, X\,) \in S_P \,\} \tag{22}$$

$$\hat{S}_P(\, B.s.t\,) = \cup_{C : (\, B, s, C\,) \in S_P} \{\, X \in 2^E : (\, C, t, X\,) \in S_P \,\} \tag{23}$$

$$\hat{S}_P(\, B.s \cap C.t\,) = \{\, X \in 2^E : (\, B, s, X\,) \in S_P \wedge (\, C, t, X\,) \in S_P \,\} \tag{24}$$

$$\hat{S}_P(\, B.s \oplus C.t\,) = \{\, X \cup Y \in 2^E : (\, B, s, X\,) \in S_P \wedge (\, C, t, Y\,) \in S_P \,\} \tag{25}$$

$$\hat{S}_P(\, B.s \otimes C.t\,) = \{\, X \cup Y \in 2^E : (\, B, s, X\,) \in S_P \wedge (\, C, t, Y\,) \in S_P \wedge X \cap Y = \phi \,\} \tag{26}$$

To prove the soundness of the credential graph, we must prove that if a path from X to $A.r$ exists in G_P, then $(A, r, X) \in S_P$. This is equivalent to showing that $\hat{S}_P(X) \subseteq \hat{S}_P(A.r)$. To show this inclusion it is sufficient to prove that $\hat{S}_P(n_1) \subseteq \hat{S}_P(n_2)$ for each edge (n_1, n_2) of G_P. This is proved in Theorem 1.

Theorem 1. For each $n_1, n_2 \in N_P$, if $(n_1, n_2) \in E_P$ then $\hat{S}_P(n_1) \subseteq \hat{S}_P(n_2)$.

Proof. Let $(n_1, n_2) \in E_P$ be an arbitrary edge in G_P. The proof is by induction with respect to the number k of steps, which are needed to add (n_1, n_2) to the constructed credential graph.

If $k = 1$, then credential $n_2 \leftarrow n_1$ must belong to P. This credential can be one of the six types allowed in RT^{T}. Each of these types will be considered separately.

$[A.r \leftarrow X]$ $(A, r, X) \in S_P$ due to (1). $\hat{S}_P(X) \subseteq \hat{S}_P(A.r)$ according to (22) above.

$[A.r \leftarrow B.s]$ Consider an arbitrary $X \in \hat{S}_P(B.s)$. This implies $(B, s, X) \in S_P$ according to (22), and $(A, r, X) \in S_P$ according to (2). Hence, $X \in \hat{S}_P(A.r)$.

$[A.r \leftarrow B.s.t]$ Consider an arbitrary $X \in \hat{S}_P(B.s.t)$. According to (23), there exists $C \in 2^E$, such that $(B, s, C) \in S_P$ and $(C, t, X) \in S_P$. This implies $(A, r, X) \in S_P$ according to (3). Hence, $X \in \hat{S}_P(A,r)$.

$[A.r \leftarrow B.s \cap C.t]$ Consider an arbitrary $X \in \hat{S}_P(B.s \cap C.t)$. This implies $(B, s, X) \in S_P$ and $(C, t, X) \in S_P$ according to (24), and $(A, r, X) \in S_P$ according to (4). Hence, $X \in \hat{S}_P(A,r)$.

$[A.r \leftarrow B.s \oplus C.t]$ Consider an arbitrary $Z \in \hat{S}_P(B.s \oplus C.t)$. According to (25), there exist $X, Y \in 2^E$ such that $X \cup Y = Z$ and $(B, s, X) \in S_P$ and $(C, t, Y) \in S_P$. Hence, $(A, r, Z) \in S_P$ according to (5), which implies that $Z \in \hat{S}_P(A,r)$.

$[A.r \leftarrow B.s \otimes C.t]$ Consider an arbitrary $Z \in \hat{S}_P(B.s \otimes C.t)$. According to (26), there exist $X, Y \in 2^E$ such that $X \cup Y = Z$ and $(B, s, X) \in S_P$ and $(C, t, Y) \in S_P$ and $X \cap Y = \phi$. Hence, $(A, r, Z) \in S_P$ according to (6), which implies that $Z \in \hat{S}_P(A,r)$.

If $k > 1$, assume as the inductive hypothesis that the thesis is true for the number of steps not greater than $k-1$. We will show that it is true also for the number of steps k. In step k one of the properties 2 through 5 in Definition 2 has been applied to add (n_1, n_2) to the constructed graph. Each of these cases will be considered separately.

Property 2. Consider a derived edge (n_1, n_2) such that $n_1 = C.t$ and $n_2 = B.s.t$. The existence of the derived edge $(C.t, B.s.t)$ in G_k implies that a path from C to $B.s$ exists in G_{k-1}. Then $\hat{S}_P(C) \subseteq \hat{S}_P(B.s)$ according to the inductive hypothesis, and $(B, s, C) \in S_P$. Consider an arbitrary $X \in \hat{S}_P(C.t)$. This implies $(C, t, X) \in S_P$ according to (22), and $X \in \hat{S}_P(B.s.t)$ according to (23).

Property 3. Consider a derived edge (n_1, n_2) such that $n_1 = X$ and $n_2 = B.s \cap C.t$. The existence of the derived edge $(X, B.s \cap C.t)$ in G_k implies that paths from X to $B.s$ and from X to $C.t$ exist in G_{k-1}. Then $\hat{S}_P(X) \subseteq \hat{S}_P(B.s)$ and $\hat{S}_P(X) \subseteq \hat{S}_P(C.t)$ according to the inductive hypothesis. Hence, $(B, s, X) \in S_P$ and $(C, t, X) \in S_P$. This implies $X \in \hat{S}_P(B.s \cap C.t)$ according to (24).

Property 4. Consider a derived edge (n_1, n_2), where $n_1 = X \cup Y$ and $n_2 = B.s \oplus C.t$. The existence of the derived edge $(X \cup Y, B.s \oplus C.t)$ in G_k implies that paths from X to $B.s$ and from Y to $C.t$ exist in G_{k-1}. Then $\hat{S}_P(X) \subseteq \hat{S}_P(B.s)$ and $\hat{S}_P(Y) \subseteq \hat{S}_P(C.t)$ according to the inductive hypothesis. Hence, $(B, s, X) \in S_P$ and $(C, t, Y) \in S_P$, which implies $X \cup Y \in \hat{S}_P(B.s \oplus C.t)$ according to (25).

Property 5. Consider a derived edge (n_1, n_2), where $n_1 = X \cup Y$ and $n_2 = B.s \otimes C.t$. The existence of the derived edge ($X \cup Y, B.s \otimes C.t$) in G$_k$ implies that $X \cap Y = \phi$ and paths from X to $B.s$ and from Y to $C.t$ exist in G_{k-1}. Then $\hat{S}_P(X) \subseteq \hat{S}_P(B.s)$ and $\hat{S}_P(Y) \subseteq \hat{S}_P(C.t)$ according to the inductive hypothesis. Hence, $(B, s, X) \in S_P$ and $(C, t, Y) \in S_P$, which implies $X \cup Y \in \hat{S}_P(B.s \oplus C.t)$ according to (26). □

To prove the completeness of the credential graph, we must prove that for each element $(A, r, X) \in S_P$ the role expressions $A.r$ and X are among the nodes of G_P and a path from X to $A.r$ exists in G_P. This is proved in Theorem 2.

Theorem 2. If $(A, r, X) \in S_P$ then $A.r$, $X \in N_P$ and a path from X to $A.r$ exists in G_P.

Proof. The proof is by induction with respect to the number k of steps to construct the element $(A, r, X) \in S_P$. Assume $(A, r, X) \in S_k$ for a certain $k \geq 1$.

If $k = 1$, then $A.r \leftarrow X \in P$, because $S_0 = \phi$ and no property other than (1) could be applied. A path from X to $A.r$ exists in the graph according to point 1 in Definition 2.

If $k > 1$, assume for the inductive step that the thesis is true up to $k-1$ steps. We will show that it is true also for k steps. In step k one of the properties (1) through (6) has been applied to construct $(A, r, X) \in S_k$. Each of these cases is discussed separately.

[$\boldsymbol{A.r \leftarrow X}$] If this is the case, then $A.r \leftarrow X \in P$, which implies that $(X, A.r) \in E_P$.

[$\boldsymbol{A.r \leftarrow B.s}$] If $A.r \leftarrow B.s$ was applied in step k to construct $(A, r, X) \in S_k$, then $(B, s, X) \in S_{k-1}$. Hence, there exists in G_P a path from X to $B.s$, according to the inductive hypothesis, and an edge from $B.s$ to $A.r$ according to point 1 in Definition 2.

[$\boldsymbol{A.r \leftarrow B.s.t}$] If $A.r \leftarrow B.s.t$ was applied in step k to construct $(A, r, X) \in S_k$, then $(B, s, C) \in S_{k-1}$ and $(C, t, X) \in S_{k-1}$. Hence, there exist in G_P paths from C to $B.s$ and from X to $C.t$ according to the inductive hypothesis, an edge (a derived edge) from $C.t$ to $B.s.t$ according to point 2 in Definition 2, and an edge from $B.s.t$ to $A.r$ according to point 1 in Definition 2. The segments: path from X to $C.t$, the derived edge from $C.t$ to $B.s.t$ and the edge from $B.s.t$ to $A.r$ comprise a path from X to $A.r$.

[$\boldsymbol{A.r \leftarrow B.s \cap C.t}$] If $A.r \leftarrow B.s \cap C.t$ was applied in step k to construct $(A, r, X) \in S_k$, then $(B, s, X) \in S_{k-1}$ and $(C, t, X) \in S_{k-1}$. Hence, there exists in G_P a path (a derived edge) from X to $B.s \cap C.t$ according to point 3 in Definition 2, and an edge from $B.s \cap C.t$ to $A.r$ according to point 1 in Definition 2.

[$\boldsymbol{A.r \leftarrow B.s \oplus C.t}$] If $A.r \leftarrow B.s \oplus C.t$ was applied in step k to construct $(A, r, X) \in S_k$, then there exist $Z, Y \in 2^E$ such that $Z \cup Y = X$ and $(B, s, Z) \in S_{k-1}$ and $(C, t, Y) \in S_{k-1}$. Hence, there exists in G_P a path (a derived edge) from $Z \cup Y$ to $B.s \oplus C.t$ according to point 4 in Definition 2, and an edge from $B.s \oplus C.t$ to $A.r$ according to point 1 in Definition 2.

[$\boldsymbol{A.r \leftarrow B.s \otimes C.t}$] If $A.r \leftarrow B.s \otimes C.t$ was applied in step k to construct $(A, r, X) \in S_k$, then there exist $Z, Y \in 2^E$ such that $Z \cup Y = X$, $Z \cap Y = \phi$ and $(B, s, Z) \in S_{k-1}$ and $(C, t, Y) \in S_{k-1}$. Hence, there exists in G_P a path (a derived edge) from $Z \cup Y$ to $B.s \otimes C.t$ according to point 5 in Definition 2, and an edge from $B.s \otimes C.t$ to $A.r$ according to point 1 in Definition 2. □

A conclusion from Theorem 1 and Theorem 2 is such that the credential graph of Definition 2 is sound and complete with respect to the semantics of RT$^{\mathrm{T}}$ credentials.

3.3 Credential Chain

A decision on whether a group of entities X is a member of role $A.r$ can be made by checking whether a path from X to $A.r$ exists in the RT^T credential graph built over the set of known credentials. The practical problem is, however, that the number of all the credentials can be very big and not all of them can be available at the moment. A solution to this problem, suggested in [13] for RT_0, is a credential chain, which is the minimal part of the credential graph that contains a path from X to $A.r$. A credential chain from X to $A.r$ is sufficient to decide on the membership of X in the role $A.r$. If such a credential chain cannot be built, the membership of X in $A.r$ is not confirmed.

A definition of RT_0 credential chain can be extended into RT^T. The chain can be built starting from one end ($A.r$) or from the other end (X), and continue the process as long as a path from X to $A.r$ is found. The resulting subgraph need not be minimal. Therefore we omit the word minimal in the definition of the credential chain.

Definition 3 (Credential Chain). A credential chain from X to $A.r$ is a subset of the credential graph containing a path from X to $A.r$ and the support sets for each derived edges in the subset. □

Let P be a set of RT^T credentials, $A.r$ be a role, and X be a set of entities from E. A credential chain is a directed graph, which nodes are role expressions that appear in credentials from P and subsets of entities from E, and directed edges reflect inclusion of sets of entities that are the meaning of those expressions.

The construction of a credential chain from X to $A.r$ can begin from node $A.r$ and proceed backward with respect to the direction of arcs, adding arcs and nodes, until the final nodes of the graph. Alternatively, the credential chain can be constructed starting from X and proceeding forward with respect to the direction of arcs.

Both algorithms are described below. Nodes that are added in the construction process can be classified into two categories: active nodes and passive nodes. Active nodes are those which initiate actions within the construction process. Passive nodes are considered only within these actions. All the active nodes represent roles.

Algorithm 1. A backward search algorithm to construct a credential chain from Z to $U.v$ consists of the following steps.

1. Create a node, which represents the role $U.v$. The created node is an active node.
2. Select an active node, denoted here $A.r$, find all the credentials $A.r \leftarrow e$, where e is an arbitrary role expression, and for each such credential do:
   a) Add node e to the set of nodes and add ($e, A.r$) to the set of edges of the constructed graph.
   b) If the credential is of type $A.r \leftarrow B.s.t$, then add $B.s$ to the set of nodes.
   c) If the credential is of type $A.r \leftarrow B.s \cap C.t$ or $A.r \leftarrow B.s \oplus C.t$ or $A.r \leftarrow B.s \otimes C.t$, then add $B.s$ and $C.t$ to the set of nodes of the constructed graph.
3. Repeat as many times as possible:
   a) If there exist nodes $B.s.t$, $B.s$ and C in the constructed graph and a path from C to $B.s$ exists in the graph, then add $C.t$ to the set of nodes and add ($C.t, B.s.t$) to the set of edges of the constructed graph.

b) If there exist nodes $B.s \cap C.t$, $B.s$, $C.t$ and X in the constructed graph such that paths from X to $B.s$ and from X to $C.t$ exist in the graph, then add (X, $B.s \cap C.t$) to the set of edges of the constructed graph.

c) If there exist nodes $B.s \oplus C.t$, $B.s$, $C.t$, X and Y in the constructed graph such that paths from X to $B.s$ and from Y to $C.t$ exist in the graph, then add $X \cup Y$ to the set of nodes and add ($X \cup Y$, $B.s \oplus C.t$) to the set of edges of the constructed graph.

d) If there exist nodes $B.s \otimes C.t$, $B.s$, $C.t$, X and Y in the constructed graph such that $X \cap Y = \phi$ and paths from X to $B.s$ and from Y to $C.t$ exists in the graph, then add $X \cup Y$ to the set of nodes and add ($X \cup Y$, $B.s \otimes C.t$) to the set of edges of the constructed graph.

4. Each node that is a role added in step 2 or 3 becomes active; all other nodes added in step 2 or 3 are passive. Node $A.r$ considered in step 2 becomes passive as well.
5. If one of the added nodes is Z, then a credential chain has been built. If the list of active nodes is empty, then a credential chain from Z to $U.v$ does not exist. Otherwise, go to step 2. □

Forward search algorithm is a bit more complex, because the starting point of the construction is not as obvious as in the previous case. To capture the problem consider a set P of three credentials, e.g.: P = { $A.r \leftarrow B.s \oplus C.t$, $B.s \leftarrow X$, $C.t \leftarrow Y$ }. It can easily be seen that the only set of entities that can jointly play the manifold role $A.r$ is the union of sets $X \cup Y$. But if we denote the union $Z = X \cup Y$, and ask about the existence of a credential chain from Z to $A.r$, then we find that the set Z does not appear within the credentials of set P. The starting point for a forward search algorithm in RT^T is then not the set Z itself, but rather the power set of Z.

Algorithm 2. A forward search algorithm to construct a credential chain from Z to $U.v$ consists of the following steps.

1. For each nonempty subset $X \subseteq Z$, find all the credentials $A.r \leftarrow X$. For each such credential, add nodes X and $A.r$ to the set of nodes and add (X, $A.r$) to the set of edges of the constructed graph. Each node that is a role added in this step becomes active; all other nodes are passive.
2. Select an active node, denoted here $B.s$, and do:

   a) Find all the credentials $A.r \leftarrow e$ such that e is a role expression $B.s$, $B.s.t$, $B.s \cap C.t$, $B.s \oplus C.t$ or $B.s \otimes C.t$ and $C.t$ is a node that exists in the constructed graph. For each such credential, add node e to the set of nodes and add (e, $A.r$) to the set of edges of the constructed graph.

   b) Find all the credentials $A.r \leftarrow B$. For each such credential, add nodes B and $A.r$ to the set of nodes and add (B, $A.r$) to the set of edges of the constructed graph.
3. Repeat as many times as possible:

   a) If there exist nodes $B.s.t$, $B.s$, C and $C.t$ in the constructed graph such that a path from C to $B.s$ exists in the graph, then add ($C.t$, $B.s.t$) to the set of edges of the constructed graph.

   b) If there exist nodes $B.s \cap C.t$, $B.s$, $C.t$ and X in the constructed graph such that paths from X to $B.s$ and from X to $C.t$ exist in the graph, then add (X, $B.s \cap C.t$) to the set of edges of the constructed graph.

   c) If there exist nodes $B.s \oplus C.t$, $B.s$, $C.t$, X and Y in the constructed graph such that paths from X to $B.s$ and from Y to $C.t$ exist in the graph, then add $X \cup Y$ to the set of nodes and add ($X \cup Y$, $B.s \oplus C.t$) to the set of edges of the constructed graph.
   d) If there exist nodes $B.s \otimes C.t$, $B.s$, $C.t$, X and Y in the constructed graph such that $X \cap Y = \phi$ and paths from X to $B.s$ and from Y to $C.t$ exist in the graph, then add $X \cup Y$ to the set of nodes and add ($X \cup Y$, $B.s \otimes C.t$) to the set of edges of the constructed graph.

4. Each node that is a role added in step 2 or 3 becomes active; all other nodes added in step 2 or 3 are passive. Node $B.s$ considered in step 2 becomes passive as well.
5. If nodes $U.v$ and Z exist in the constructed graph, then a credential chain has been built. If the list of active nodes is empty, then a credential chain from Z to $U.v$ does not exist. Otherwise, go to step 2. □

3.4 An Example

Consider the following question: Can a team of *Adam* and *Betty* approve a big transaction in the bank from the example in Section 2.3?

To answer this question we can construct a credential chain from {*Adam*, *Betty*} to {*Bank*}.*approveBig* (Fig. 1). The edges marked with a solid line are *credential edges*. Dashed lines represent the *derived* elements, which were added according to properties 2 through 5 in Definition 2.

A description of the application of backward and forward search algorithms shown below, is based on an assumption that credential (16) is used and the credentials (14) and (15) are excluded from the set P of available credentials.

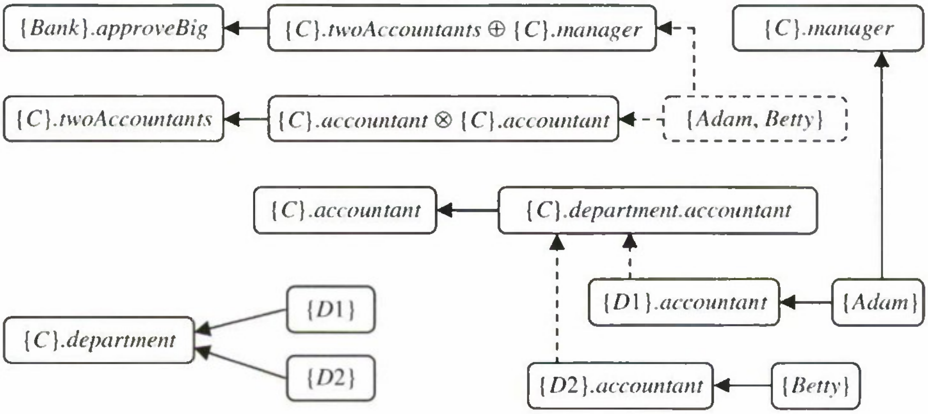


Fig. 1. Credential chain from a team of entities {*Adam*, *Betty*} to the role {*Bank*}.*approveBig*

Backward search algorithm starts at node {*Bank*}.*approveBig*. Credential (20) adds node {*C*}.*twoAccountants* ⊕ {*C*}.*manager*, according to point 2a in Algorithm 1, and nodes {*C*}.*twoAccountants* and {*C*}.*manager*, according to point 2c. Starting at

$\{C\}.\mathit{manager}$, credential (9) adds node $\{\mathit{Adam}\}$, and starting at $\{C\}.\mathit{twoAccountants}$, credential (19) adds nodes $\{C\}.\mathit{accountant} \otimes \{C\}.\mathit{accountant}$ and $\{C\}.\mathit{accountant}$.

The only active node is now $\{C\}.\mathit{accountant}$. Credential (16) adds nodes $\{C\}.\mathit{department.accountant}$ and $\{C\}.\mathit{department}$, according to 2b in Algorithm 1. Credentials (7) and (8) add nodes $\{D1\}$ and $\{D2\}$. The edges from $\{D1\}$ and $\{D2\}$ to $\{C\}.\mathit{department}$ create the support sets for derived edges from $\{D1\}.\mathit{accountant}$ and $\{D2\}.\mathit{accountant}$ to $\{C\}.\mathit{department.accountant}$, added according to point 3a.

Next, credentials (11) and (13) add edges from $\{\mathit{Adam}\}$ to $\{D1\}.\mathit{accountant}$ and from $\{\mathit{Betty}\}$ to $\{D2\}.\mathit{accountant}$. According to point 3d in Algorithm 1, a derived node $\{\mathit{Adam}, \mathit{Betty}\}$ is created and added to the graph, together with a derived edge to $\{C\}.\mathit{accountant} \otimes \{C\}.\mathit{accountant}$. Finally, a derived edge from $\{\mathit{Adam}, \mathit{Betty}\}$ to $\{C\}.\mathit{twoAccountants} \oplus \{C\}.\mathit{manager}$ is added, according to point 3c in Algorithm 1.

The path from $\{\mathit{Adam}, \mathit{Betty}\}$ to $\{\mathit{Bank}\}.\mathit{approveBig}$ shows that this team of entities can approve a big transaction according to the security policy of the bank.

Forward search algorithm starts at sets of entities $\{\mathit{Adam}\}$, $\{\mathit{Betty}\}$ and $\{\mathit{Adam}, \mathit{Betty}\}$. Credentials (9), (11), (13) add nodes $\{\mathit{Adam}\}$, $\{\mathit{Betty}\}$, $\{C\}.\mathit{manager}$, $\{D1\}.\mathit{accountant}$ and $\{D2\}.\mathit{accountant}$ to the constructed graph, and add edges from $\{\mathit{Adam}\}$ to $\{C\}.\mathit{manager}$, from $\{\mathit{Adam}\}$ to $\{D1\}.\mathit{accountant}$ and from $\{\mathit{Betty}\}$ to $\{D2\}.\mathit{accountant}$. Moreover, credentials (11), (13) and {7}, {8} add nodes $\{D1\}$, $\{D2\}$ and $\{C\}.\mathit{department}$, and add edges from $\{D1\}$ to $\{C\}.\mathit{department}$ and from $\{D2\}$ to $\{C\}.\mathit{department}$.

Next, credential (16) adds nodes $\{C\}.\mathit{department.accountant}$ and $\{C\}.\mathit{accountant}$, and adds an edge from $\{C\}.\mathit{department.accountant}$ to $\{C\}.\mathit{accountant}$. The active node is now $\{C\}.\mathit{accountant}$. Credential (19) adds nodes $\{C\}.\mathit{twoAccountants}$ and $\{C\}.\mathit{accountant} \otimes \{C\}.\mathit{accountant}$ and the edge between the two. Moreover, a node $\{\mathit{Adam}, \mathit{Betty}\}$ is created together with an edge to $\{C\}.\mathit{accountant} \otimes \{C\}.\mathit{accountant}$.

Finally, credential (20) creates two nodes $\{C\}.\mathit{twoAccountants} \oplus \{C\}.\mathit{manager}$ and $\{\mathit{Bank}\}.\mathit{approveBig}$, and two edges from $\{C\}.\mathit{twoAccountants} \oplus \{C\}.\mathit{manager}$ to $\{\mathit{Bank}\}.\mathit{approveBig}$ and from $\mathit{Adam}, \mathit{Betty}\}$ to $\{C\}.\mathit{twoAccountants} \oplus \{C\}.\mathit{manager}$.

A credential chain from $\{\mathit{Adam}, \mathit{Betty}\}$ to $\{\mathit{Bank}\}.\mathit{approveBig}$ has been built.

4 Conclusions

Role-based Trust management languages use credentials to define security policies and handle trust in decentralized distributed access control systems. RT^{T} is a powerful language, which supports manifold roles and is capable of expressing threshold and separation of duties policies. The contribution of this paper is a modified definition of the relational RT^{T} semantics and definitions of RT^{T} credential graph and credential chain, which allow searching a given set of credentials and answering the security queries. The soundness and completeness of the credential graph with respect to the semantics of RT^{T} is proved.

The plans for further research include construction of a prototype implementation of a trust management server. Neither the existing implementations of the trust management systems [1,2,3,5] nor the development described in the literature [14] are able to use the potential of RT^{T} language. Our ultimate goal is an environment, in which access control is one of the services offered in the system (Fig. 2).

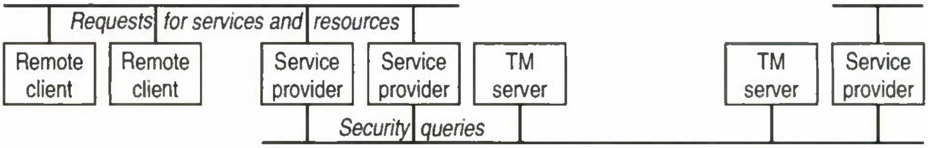


Fig. 2. Trust management (TM) server in a service-oriented environment

References

1. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized Trust Management. In: 17th IEEE Symposium on Security and Privacy, pp. 164–173. IEEE Computer Society Press, Los Alamitos (1996)
2. Blaze, M., Feigenbaum, J., Strauss, M.: Compliance Checking in the PolicyMaker Trust Management System. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 254–274. Springer, Heidelberg (1998)
3. Blaze, M., Feigenbaum, J., Ioannidis, J.: The KeyNote Trust Management System Version 2. Internet Society, Network Working Group, RFC 2704 (1999)
4. Chapin, P., Skalka, C., Wang, X.: Authorization in Trust Management: Features and Foundations. ACM Comput. Survey 3, 1–48 (2008)
5. Clarke, D., Elien, J.-E., Ellison, C., Fredette, M., Morcos, A., Rivest, R.L.: Certificate chain discovery in SPKI/SDSI. J. Computer Security 9, 285–322 (2001)
6. Czenko, M., Etalle, S., Li, D., Winsborough, W.: An Introduction to the Role Based Trust Management Framework RT. In: Aldini, A., Gorrieri, R. (eds.) FOSAD 2007. LNCS, vol. 4677, pp. 246–281. Springer, Heidelberg (2007)
7. Felkner, A., Sacha, K.: The Semantics of Role-Based Trust Management Languages. In: 4th IFIP Central and East European Conference on Software Engineering Techniques, pp. 195–206 (2009)
8. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramouli, R.: Proposed NIST Standard for Role-Based Access control. ACM Trans. Inf. Syst. Secur. 3, 224–274 (2001)
9. Gorla, D., Hennessy, M., Sassone, V.: Inferring Dynamic Credentials for Role-Based Trust Management. In: 8th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, pp. 213–224. ACM, New York (2006)
10. Harel, D., Rumpe, B.: Modeling Languages: Syntax, Semantics and All That Stuff, Part I: The Basic Stuff. Weizmann Science Press of Israel, Jerusalem (2000)
11. Li, N., Mitchell, J., Winsborough, W.: Design of a Role-Based Trust-Management Framework. In: IEEE Symposium on Security and Privacy, pp. 114–130. IEEE Computer Society Press, Los Alamitos (2002)
12. Li, N., Mitchell, J.: RT: A Role-Based Trust-Management Framework. In: 3rd DARPA Information Survivability Conference and Exposition, pp. 201–212. IEEE Computer Society Press, Los Alamitos (2003)
13. Li, N., Winsborough, W., Mitchell, J.: Distributed Credential Chain Discovery in Trust Management. J. Computer Security 1, 35–86 (2003)
14. Reith, M., Niu, J., Winsborough, W.: Engineering Trust Management into Software Models. In: International Workshop on Modeling in Software Engineering. IEEE Computer Society, Los Alamitos (2007)
15. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-Based Access Control Models. IEEE Computer (2), 38–47 (1996)

Genetic Optimization of Access Control Schemes in Virtual Local Area Networks

Igor Saenko and Igor Kotenko

St. Petersburg Institute for Informatics and Automation (SPIIRAS)
39, 14 Linija, St. Petersburg, Russia
{saenko,ivkote}@comsec.spb.ru

Abstract. The paper presents the formulation of the problem of access control to information resources located in virtual local area networks. We define the initial data, the objective function and constraints of the problem. To solve the proposed problem we suggest the method of genetic optimization of access control scheme based on the poly-chromosomal representation of intermediate points. The results of computer simulation and evaluation of the proposed method are discussed.

Keywords: access control, virtual local area networks, genetic optimization.

1 Introduction

Joint work of users in computer networks stipulates the need to restrict the access to information resources without the use of passwords. An example is the problem of protecting information from unauthorized access in computer classrooms of universities.

This problem has the following specificity. First, the student contingent has a strong heterogeneity, and all students can be considered as potential security infringers. In this case, the effectiveness of passwords and user accounts is low. Secondly, in classrooms the access control schemes require frequent retuning. This is due to the fact that in classrooms the lessons, having different composition of used information resources, are usually alternated.

The basic principles of information security in such integrated information systems are outlined, for instance, in the papers [1, 2]. These papers show that the discretionary model based on an access control matrix is most widely implemented in classrooms of universities. The first access control matrix as access control scheme was introduced in [3]. This model was considered in more details in many modern works, for example [4, 5]. Each cell of the matrix defines the subject authority to access a specific object or another access subject.

In practice, as a rule, the access control matrix is replaced by access control lists (ACL) [6] or "lists of capabilities" (C-lists) [7]. Switches, used for local area networks, also use ACL [8]. Such capability allows to implement virtual local area networks (VLANs) based on these solutions [9]. ACL lists ensure that certain traffic is sent to specific ports. This prevents the unauthorized access to confidential corporate

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 209–216, 2010.

information and network congestion as a result of program attacks. As a result, on the one hand, VLANs provide an additional level of access control to network resources, and, on the other hand, the adjustment of VLANs is also determined by the access control matrix.

The generation of an access control matrix is a complex problem [10]. Under system operation the adjustment of access control schemes is repeated each time, when the equipment, software and users are changed. Nevertheless, usually the generation of access control scheme is still done manually, without the use of mathematical methods [11]. Creation of access control scheme can be automated, if we reduce it to an optimization problem and apply an effective way to solve it. One of these ways is to use genetic algorithms. Genetic algorithms allow to solve successfully the problems of structural and parametric optimization of various systems [12, 13].

The purpose of this paper is to test the idea of applying genctic algorithms to generate a correct access control matrix for a computer network on the base of constructing VLAN. We suggest the method of genetic optimization of access control scheme based on the poly-chromosomal representation of intermediate points.

The paper is structured as follows. *Section 2* considers mechanisms for access control to information in VLAN. *Section 3* outlines the proposed problem definition and analysis. In *section 4*, we suggest the method of genetic optimization of access control scheme. *Section 5* discusses the results of computer simulation and evaluation of the proposed method. *Conclusion* surveys the main paper results.

2 Mechanisms for Access Control to Information in VLAN

The efficient mechanisms for access control and protection of information against unauthorized access in VLAN are (1) the rational distribution of information resources and users on network nodes and (2) the organization of virtual subnets using network switches or routers. Let us consider these mechanisms.

2.1 Distribution of Information Resources and Users on Network Nodes

Information resources (files or directories) distributed on network nodes are called *access objects*. Network computers are *network nodes*. Users, working at computers at any given time, are called *access subjects*.

Several access objects can be situated on one network node at one time. In other words, there is a mapping $\mathbf{D}^{OU}$ of degree 1 : M among the set of access objects and the set of nodes. The same access subject can work only on one node. Consequently, the mapping $\mathbf{D}^{SU}$ among the set of subjects and the set of nodes has also the degree 1 : M.

Access subjects have full access to those objects which are located on their own network node. At the same time, sometimes, the subject has to access one or more objects on other nodes (for example, on a network server). This capability is achieved by assigning to access object a special *shared access flag*. It should be noted that there is the opportunity of a password based shared access. However, this type of shared access is not taken into account in the statement of the problem due the specificity considered in the Introduction. The password based shared access control is assigned to VLAN.

It is supposed that the access of a specific subject to a particular object is determined by the following rules: (1) access is possible, if the object does not have a shared access flag, but is located on the node at which the subject operates; (2) access is possible, if the object has that flag (in this case it is not important on which node the object is located); (3) access is denied in all other cases.

2.2 Organization of Virtual Subnets

Virtual subnets are realized by using managed network switches. These network devices have a memory that stores information on banning (permitting) the exchange of information between certain pairs of computers connected to switches. As a result, it is possible instead of a fully connected exchange scheme between the ports to organize a selective scheme with segregation of virtual local subnets.

The following access rule is used in VLAN for all computers: if two computers are not in the same subnet, then the information exchange between them is impossible.

VLAN implementation requires a change of the second rule outlined above. Now this rule is as follows: access is possible, if the object has a shared access flag and the computer, on which the object is located, and the computer, on which the subject works, are in the same virtual subnet.

The simultaneous use of two considered access control mechanisms makes up a "real" access control scheme. At the same time the usage the specific software determine a "required" access control scheme. In the general case a "real" and a "required" access control schemes may be different.

Thus, an informal statement of the problem of access control with usage of VLAN can be formulated as follows: using mentioned access control mechanisms it is needed to ensure that the "real" access control scheme has minimal differences with the "required" scheme, and coincides with it in ideal case.

3 Formal Statement of the Problem

Let us specify the formal statement of the problem of on-line optimization of access control schemes in VLANs.

The initial data for the formal statement of the problem are as follows:

OD = $\{od_i\}$, $i = 1...I$ – set of access objects (files, directories);

SD = $\{sd_j\}$, $j = 1...J$ – set of access subjects (for example, learners and teachers working in a computer network);

U = $\{u_k\}$, $k = 1...K$ – set of network nodes;

$\mathbf{R}^{\mathrm{req}} = \|r^{\mathrm{req}}_{ij}\|$ – requirements for different levels of access (required access control scheme), where $r^{\mathrm{req}}_{ij} = 1$, if sd_j should have access od_i, and $r^{\mathrm{req}}_{ij} = 0$ otherwise.

Since the problem variables should fully determine the decisions on the distribution of objects and subjects and the structure of VLAN, we assume that these decisions are as follows:

$\mathbf{D}^{\mathrm{OU}} = \|d^{\mathrm{OU}}_{ik}\|$ – matrix of distribution of objects on network nodes, where $d^{\mathrm{OU}}_{ik} = 1$, if od_i is located on the node u_k, and $d^{\mathrm{OU}}_{ik} = 0$ otherwise;

$\mathbf{D}^{\mathrm{SU}} = \|d^{\mathrm{SU}}_{jk}\|$ – matrix of distribution of subjects on network nodes, where $d^{\mathrm{SU}}_{jk} = 1$, if sd_j is located on the node u_k, and $d^{\mathrm{SU}}_{jk} = 0$ otherwise;

$\mathbf{V} = \{v_i\}$ – vector of shared access flags of network resources, where $v_i = 1$, if od_i is given in the share, and $v_i = 0$ otherwise;

$\mathbf{X} = \|x_{mn}\|$, $m,n = 1 \ldots K$ – matrix of VLAN structure, where $x_{mn} = 1$, if nodes u_m and u_n belong to one virtual subnet, and $x_{mn} = 0$ otherwise.

As an objective function should be used the function, evaluating the difference between the real control access scheme $\mathbf{R}^{\text{real}}$, stipulated by values $\mathbf{D}^{\text{OU}}$, $\mathbf{D}^{\text{SU}}$, $\mathbf{V}$ and $\mathbf{X}$, and the required access scheme $\mathbf{R}^{\text{req}}$.

Let us show how to obtain the functional form of scheme $\mathbf{R}^{\text{real}}$.

Assume that the access control scheme is determined only by the decisions $\mathbf{D}^{\text{OU}}$ and $\mathbf{D}^{\text{SU}}$ (in other words, all the elements of $\mathbf{V}$ and $\mathbf{X}$ are equal to 1). We call this scheme unconditional and denote $\mathbf{R}^{\text{uc}}$. In this case we have

$$\mathbf{R}^{\text{uc}} = \mathbf{D}^{\text{SU}} \cdot (\mathbf{D}^{\text{OU}})^{\text{T}}, \tag{1}$$

where the elements of the matrix $\mathbf{R}^{\text{uc}}$ are determined by the expression

$$r^{\text{uc}}{}_{ij} = \sum_{k=1}^{K} \left(d^{\text{SU}}{}_{ik} \cdot d^{\text{OU}}{}_{kj} \right). \tag{2}$$

Note that in (2), as in all subsequent expressions, summation and product are the logical operators OR and AND respectively.

Suppose that the decision $\mathbf{V}$ takes effect (that is, there are v_i=0). We call this access control scheme as "conditional on $\mathbf{V}$" and denote $\mathbf{R}^{\text{V}}$.

If $v_i = 1$, then the resource od_i is available for all subjects. In this case, for any j, $r^{\text{V}}{}_{ij} = 1$. If $v_i = 0$, then the availability of the resource od_i is defined by a matrix $\mathbf{R}^{\text{uc}}$. Consequently, the element of the matrix $\mathbf{R}^{\text{V}}$ is defined by the following expression

$$r^{\text{V}}{}_{ij} = r^{\text{uc}}{}_{ij} + v_i\,(1 - r^{\text{uc}}{}_{ij}). \tag{3}$$

Now suppose that in addition to $\mathbf{V}$, the decision $\mathbf{X}$ enters into force. In this case, the access control scheme is a "real" control access scheme $\mathbf{R}^{\text{real}}$.

The actual availability of the resource od_i to subject sd_j occurs when there is a virtual subnet joining this resource and this subject together. In other words, the following expression is true:

$$r^{\text{real}}{}_{ij} = \sum_{k=1}^{K} x_{ik} \cdot r^{\text{V}}{}_{kj}. \tag{4}$$

It is easy to see that expressions (2)–(4) completely determine $\mathbf{R}^{\text{real}}$ as a function of variables $\mathbf{D}^{\text{OU}}$, $\mathbf{D}^{\text{SU}}$, $\mathbf{V}$ and $\mathbf{X}$.

Objective function of optimization problem statement is defined as a measure of divergence between $\mathbf{R}^{\text{real}}$ and $\mathbf{R}^{\text{req}}$

$$\Delta\mathbf{R} = \sum_{i=1}^{I} \sum_{j=1}^{J} \left| r^{\text{real}}{}_{ij} - r^{\text{req}}{}_{ij} \right|. \tag{5}$$

The discrepancy between $\mathbf{R}^{\text{real}}$ and $\mathbf{R}^{\text{req}}$ should be minimal. Therefore, the synthesis criterion formulated in the problem statement has the form

$$\Delta\mathbf{R}\left(\mathbf{D}^{\text{OU}}, \mathbf{D}^{\text{SU}}, \mathbf{V}, \mathbf{X}, \mathbf{R}^{\text{req}}\right) \Rightarrow \min . \tag{6}$$

The constraints of the problem statement are as follows:

1) on a single node there can not be more than one subject, and therefore the following condition is true:

$$\sum_{k=1}^{K}\left(d^{\text{SU}}{}_{ik}\right) \leq 1 ; \tag{7}$$

2) one file can be only on one node, so the following expression is valid

$$\sum_{k=1}^{K}\left(d^{\text{OU}}{}_{ik}\right) \leq 1 . \tag{8}$$

4 Method of Solving the Problem

The problem defined by expressions (2) – (8) belongs to a class of non-linear Boolean programming problems, when the variables are given in the vector and matrix form. The exact solution of this problem is possible only by an exhaustive search of variables that can not be acceptable for practical purposes.

We offer for its solution a method which implements genetic optimization algorithms (GOA), successfully used in many synthesis problems [12, 13].

However, we note that, as shown by expression (3) and (4), the set of variables in the objective function (6) can be reduced by replacing the two matrices $\mathbf{D}^{\text{OU}}$ and $\mathbf{D}^{\text{SU}}$ on a single matrix $\mathbf{R}^{\text{uc}}$.

The method based on GOA is as follows. On initialization stage, an initial set of solutions (or *population*) is randomly formed. Each solution (or *individual*) is characterized by a string isomorphically related to the vectors and matrices of variables that determine this solution. This string is called a *chromosome* and a single character in it – a *gene*.

At each subsequent stage the following steps are fulfilled.

Pairs from the population of individuals are randomly selected. They are called *parents*. Between them the process of *crossing-over* occurs. As a result of this process, a couple of new individuals appear. These individuals are called *descendants*. The chromosome of each of the descendants is formed from two parts: one part is taken from the chromosomes of the "father", and the second – from the "mother"'s chromosomes. The descendants are added to the general population.

The population has quantitative restrictions, so individuals with the lowest suitability function are removed from the population ("die"). The role of suitability function is played by the function (5).

In addition, at each stage a part of the individuals is subjected to *mutation*. During mutation the genes in the chromosome are changed randomly.

An essential feature of proposed GOA is his poly-chromosomal character, i.e. individuals have not one, but three chromosomes $\mathbf{R}^{uc}$, **V** и **X**.

Let us offer the forms of these chromosomes.

Since $\mathbf{R}^{uc}$ is a matrix of dimension I x J, it is not symmetric. Therefore, the only way to build a chromosome mapping this matrix is a serial concatenation of rows of $\mathbf{R}^{uc}$ into one big string:

$$[\mathbf{R}]_{\mathrm{chr}} = [r_{11}, \ldots, r_{1J}; x_{21}, \ldots, x_{2J}; \ldots; x_{i1}, \ldots, x_{iJ}; \ldots; x_{I1}, \ldots, x_{IJ}]. \tag{9}$$

Vector **V** by its very nature is a chromosome, in which an element v_i carries the role of individual gene:

$$[\mathbf{V}]_{\mathrm{chr}} = [v_1, v_2, \ldots, v_i, \ldots, v_I]. \tag{10}$$

Matrix **X** is a symmetric matrix. Each element of its main diagonal is 1. Therefore, to construct the chromosome which maps **X**, the following string is used:

$$[\mathbf{X}]_{\mathrm{chr}} = [x_{12}, \ldots, x_{1K}; x_{23}, \ldots, x_{2K}; \ldots; x_{i,i+1}, \ldots, x_{iK}; \ldots; x_{K-1,K}]. \tag{11}$$

As a result of poly-chromosomal crossing-over, not two, as in the traditional case, but eight descendants ($2^3 = 8$) will appear.

The GOA is completed, when the population goes to a stable state, in which the individual with the maximum value of efficiency is taken as the final solution of the problem.

5 Evaluation of the Method

The evaluation was conducted in two phases. On the first phase, we estimated computational complexity and performance. On the second phase, we estimated the LAN security based on the method developed.

Analysis shows that GOA has a polynomial computational complexity $O\,(N_{\mathrm{pop}}{\cdot}N_{\mathrm{ind}}{\cdot}K)$, where N_{pop} – number of populations needed to obtain a solution, N_{ind} – number of individuals in the population, K – number of network nodes. In the experiments, the value of N_{po} was in the interval [25; 100], K had values {5; 10; 15} and $N_{\mathrm{ind}} = 200$.

Evaluation of GOA performance demonstrated that a full coincidence of the resulting access scheme with the required one is observed only at small values of I, in particular, when $I = 6$. Moreover, the coincidence is reached at population number in the range from 25 to 30.

Data on security evaluation are given in Table 1.

Table 1. Security evaluation

I	P_0	p_{pw}	N_1	N_2	P_1	P_2	k_{UUD}
6	10^{-4}	10^{-4}	6	0	0,00070	10^{-4}	7,00
6	10^{-5}	10^{-4}	6	0	0,00061	10^{-5}	60,98
6	10^{-4}	10^{-5}	6	0	0,00016	10^{-4}	1,60
12	10^{-4}	10^{-4}	12	5	0,00130	0,00060	2,17
12	10^{-5}	10^{-4}	12	5	0,00121	0,00051	2,37
12	10^{-4}	10^{-5}	12	5	0,00022	0,00015	1,47
20	10^{-4}	10^{-4}	20	18	0,00210	0,00190	1,11
20	10^{-5}	10^{-4}	20	18	0,00201	0,00180	1,11
20	10^{-4}	10^{-5}	20	18	0,00030	0,00028	1,07

The table 1 uses the following parameters: P_0 – the probability of unauthorized access the information caused by other reasons other than the compromise of shared passwords; p_{pw} – the probability of password compromising; $N1$ and $N2$ – the number of objects which require access password protection in the traditional case and in the case of using the proposed method, respectively; P_1 и P_2 – the probability of unauthorized access in the traditional case and in the case of using the proposed method, respectively; $k_{UUD} = P_1 / P_2$ – degree of security increase.

Table 1 shows that for various configurations of the simulated system the gain in security increase varies from 7 to 600 percentages. The greatest gain in 60 times takes place only when p_{pw} is greater than P_0 in 10 times, and the simulated system has a low dimension, when the resulting access scheme, organized by means of VLANs, is the same as required one. In all other cases, when the probability of compromising the password is much more than the probability of unauthorized access by other reasons, the gain is also significant.

6 Conclusion

The paper shows that combining the technologies of VLAN and GOA can be an effective means of protecting information against unauthorized access to the information stored in local networks. On the one hand, the proposed method of protecting information from unauthorized access takes into account the requirements of security policy. On the other hand, it provides multi-level use of organizational and technical measures of protection. The method has high efficiency and improves the security on 7–11 percentages for large-scaled systems or in 7–60 times for small systems. In this case the network performance was not reduced significantly, and the cost of routine work of security administrators was greatly decreased.

Software implementation of proposed method may be included in the arsenal of information security means available to network security administrators. It may be actively used to create dynamically configurable schemes of custom access to network resources.

Acknowledgments

This research is partly funded by the EU under SecFutur project, the grant of the Russian Foundation of Basic Research (Project No.10-01-00826) and Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (Contract No.3.2).

References

1. Bubnov, R.V., Chernikov, A.S.: Basic principles of security in an integrated information system to support the management of the University. Vestnik MSTU Bauman, No.2 (2004) (in Russian)
2. Simonenko, S.N.: Review of discretionary access control mechanisms in relation to information systems, http://www.philippovich.ru/Library/Books/ITS/wwwbook/IST7/simonenko/Simonenko.htm#_ftn1 (in Russian)
3. Lampson, B.W.: Protection. In: Proceedings of the 5th Princeton Conference on Information Sciences and Systems (1971)
4. Bishop, M.: Computer Security: art and science. Pearson Education, Inc., Boston (2002)
5. Kizza, J.M.: Computer Network Security. Springer Science+Business Media, Inc., New York (2005)
6. Galatenko, V.A.: Identification and authentication, access control, http://www.citforum.ru/security/articles/galatenko (in Russian)
7. Gyikovich, V.Y.: Fundamentals of Information Technology Security, http://bezpeka.ladimir.kiev.ua/pq/show/zi.htm (in Russian)
8. Hildebrandt, W.: Security at all levels. LAN 11 (2004) (in Russian)
9. Research Report: Secure Use of VLANs: An @stake Security Assessment (2002)
10. The main protective mechanisms used in systems to protect information, http://asher.ru/security/book/its/07 (in Russian)
11. How To Design an Access Control Matrix for Your Organization, http://www.howtodothings.com/business/how-to-design-an-access-control-matrix-for-your-organization
12. Wang, G., Dexter, T.W., Punch, V.F., Goodman, E.D.: Optimization of GA and Within a GA for a 2-Dimensional Layout Problem. In: First International Conference on Evolutionary Computation and Its Application (1996)
13. Shaffer, J.D., Eshelman, L.J.: Combinatorial Optimization by Genetic Algorithms: The Value of the Genotype/Phenotype Distinction. In: First International Conference on Evolutionary Computation and Its Application (1996)

Intellectual Intrusion Detection with Sequences Alignment Methods

Yaroslav A. Markov and Maxim O. Kalinin

Information Security Center of St. Petersburg Polytechnical University,
Polytehnicheskaya str. 29, St. Petersburg, Russia
{markov,max}@ssl.stu.neva.ru

Abstract. The paper addresses to application of sequences alignment intellectual algorithms for the intrusion detection needs. These algorithms are used in bioinformatics to detect regions of similarity in several gene sequences. We propose two techniques of their utilization. Using the first technique it is possible to detect the mutations of attack, having a signature of it. The second technique is applicable to anomaly detection. We discuss what algorithms of sequences alignment can be used in these methods and show the effectiveness of these techniques on practice.

Keywords: security, intrusion detection, sequences alignment, mutations of attack, anomaly detection.

1 Introduction

Development of information technologies causes the enriching of the intruder's potential. S(he) can now adapt to new detection algorithms and invent new types of attacks. It resembles the game. In a response to new methods of detection the intruder invents new methods of attacks. When the method of detection of this new attack is invented the intruder invents other new attack and etc. The main thing of it is that the number of attacks grows exponentially. These facts obligate us to improve the methods and algorithms used in IDS. The IDS, the intrusion detection system, is a software-based or hardware-based tool developed to detect malicious activities or policy violations such as unauthorized access, integrity violation or denial of service.

IDS usually solves the problem of matching two sequences to determine their likeness. For example, it can be a sequence of system calls or sequence of network packets that is compared to attack signature. In bioinformatics, this problem is solved by sequences alignment algorithms. Thus, resemblance of two problems makes research of application of these algorithms urgent for information security tasks.

The IDSs are divided in two classes: signature-based and anomaly-based. One of the problems of signature-based IDS is the problem of attack mutation. If attack is slightly changed, it can avoid IDS, and consequently a new signature for it should be developed. We show how the sequences alignment algorithms can be used to solve this problem. One of the approaches for anomaly-based systems is to create the base of normal behavior of protected component and then compare the monitored behavior

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 217–228, 2010.

with it. The base of normal behavior can be built of sequence of system calls or network packets. The sequences alignment algorithms can improve this method by reduction of the size of normal behavior base.

The following paper is divided in 7 sections. Section 2 determines the formal model of attack detection. In section 3 the sequences alignment algorithms are reviewed. Section 4 describes the application of these algorithms. Section 5 and 6 contain results and review of the related works. Finally, there is the conclusion in section 7.

2 Formal Model of Attack Detection

We define the protected system *System* as a set of entities *E* that interact with each other. Whether the interaction is permitted or denied depends on the security attributes *SA:*

$$System = <E, SA> . \tag{1}$$

Informational interaction is an interaction process of two or more entities with the purpose of changing the information within at least one of them. In any informational interaction, there are entities that initiate it. For example, a man initiates reading of a book. We denote entities that cannot be initiators of any interaction as objects *O*. All other entities we denote as subjects *S*. Thus, a set of entities is unification of *S* and *O*: $E = S\,Y\,O$. Informational interaction between the entities in the given system is implemented by executing commands which make the set *C*. The command can be written in pseudo-language as:

$$(Condition_1) \rightarrow (Inter_1, Inter_2, ..., Inter_n); (Condition_2) \rightarrow (Inter_m) . \tag{2}$$

Where $Inter_1$, $Inter_2$, ..., $Inter_n$, $Inter_m$ are the elements of possible interactions in the system set *Inter*. *Condition1*, *Condition2* are the conditions like "if *User1* has a right to read file *Documents*".

Let *AC* denotes an access control function:

$$AC:(S,O,Inter) \rightarrow \{0,1\} . \tag{3}$$

It checks whether the subject *S* can have an interaction *Inter* with the object *O*. For example, in systems that use Harrison-Russo-Ullman security model, the access control function equals to 1 iff there is *Inter* in the access matrix cell respective to *S* and *O* [1].

The condition in command is a unity or conjunction of access control functions:

$$Condition = \begin{cases} 1 \\ AC(S_1,O_1,Inter_1) \wedge AC(S_2,O_2,Inter_2) \wedge ... \wedge AC(S_n,O_n,Inter_n) \end{cases} . \tag{4}$$

State denotes a system state function which returns the tuple $<E_t, SA_t>$, where E_t and SA_t are the sets of the system entities and security attributes fixed at time t:

$$State{:}T \rightarrow < E, SA > . \tag{5}$$

Where T is a set of time moments with given discrete frequency.

IsSecure is a system security function:

$$IsSecure{:} < E, SA > \rightarrow \{0, 1\} . \tag{6}$$

IsSecure is equal to unity iff the system is secure.

The system state changes under influence of the commands. Thus, knowledge about that at the time period $[t_1, t_n]$ commands $C_1, C_2, \ldots, C_{n-1}$ were executed in the given sequence leads us to fact that the initial state at the time moment t_1 can define all states of the system for this time interval:

$$State(t_1) \xrightarrow{C_1} State(t_2) \xrightarrow{C_2} State(t_3) \xrightarrow{C_3} \ldots \xrightarrow{C_{n-1}} State(t_n) . \tag{7}$$

The insecure sequence of commands is defined as a sequence of commands $C_1, C_2, \ldots, C_{n-1}$ executed at time interval $[t_1, t_n]$ having the following conditions true:

$$\begin{gathered} State(t_1) \xrightarrow{C_1} State(t_2) \xrightarrow{C_2} State(t_3) \xrightarrow{C_3} \ldots \xrightarrow{C_{n-1}} State(t_n) , \\ (\prod_{i=1}^{n-1} IsSecure(State(t_i)) = 1) \wedge IsSecure(State(t_n)) = 0) \cdot \end{gathered} \tag{8}$$

Any influence that can be represented as a sequence of commands that bring system to a state in which *IsSecure* function is zero we will name as an 'attack on the system'. At the same time, removal of first command in attack sequence brings the *IsSecure* function to unity. Thus, the following statement is true:

The attack is always a member of insecure sequence of commands set.

As defined above, any attack fits the definition of insecure sequence of commands. The opposite statement is false.

Thus, the lifecycle of any system can be presented as a chain of commands that lead to system state change. And the attack detection problem is equal to matching these chains to known attack signatures.

3 Sequences Alignment Algorithms

In bioinformatics, a sequences alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences [2].

There are several sequences alignment algorithms and their results are different. The commonly known gene sequences alignment algorithms are local and global alignments. Lets explain their work on the following sample: there are two sequences of the commands «*open, read, open, read, write, execute, connect, execute, execute, write, close, write, close*» and «*open, read, write, execute, execute, execute, write, write, close*».

Local algorithm:

open read open read write execute connect execute execute - write close write close

- - open read write execute - execute execute write write close - -

Global algorithm:

open read open read write execute connect execute execute write close write close

open - - read write execute - execute execute write - write close

where the "-" symbol denotes a gap, i.e. absence of the command.

Global alignment which is known as the Needlman-Wunsch algorithm [3] stretches the smaller sequence along the bigger one. Local alignment which is also known as the Smith-Waterman algorithm [4] localizes the smaller sequence on the specified region of the bigger one. Both algorithms can be used on sequences of any length, but the Needleman-Wunsch algorithm is traditionally used much more often when sequences have approximately equal lengths. The Smith-Waterman algorithm is used when one sequence is considerably larger than another.

3.1 Smith-Waterman Algorithm

The algorithm's input is represented with two sequences a = «$C_{a1},C_{a2}...C_{an}$» and b = «$C_{b1},C_{b2}...C_{bm}$» and the similarity function $\omega:(C_a \cup -, C_b \cup -,) \rightarrow Z$, where C_a and C_b are the sets of the commands that form the sequences a and b respectively. The "-" symbol denotes absence of command. Target of that function is to define a similarity degree between two commands if they stand on the same positions in different sequences. For example, there is an attack with a purpose of changing some files. In that case, important commands are: gaining access rights to open and write file; opening the file, and writing into the file. For any important commands ω has to be positive in case of same arguments; and negative in case of different arguments. For any non-important commands, function ω is zero. Also, the important commands can be differentiated by danger degrce for the protected system. For example, if there is a command that deletes all entities in the system, the ω function calculated for this command can be set to 10. And for command that changes any entity in the system, the ω function can be set to 2.

Similarity degree between two sequences is represented by R function:

$$R(a,b) = \sum_{i=0}^{n-1} \omega(C_{ai}, C_{bi}) \cdot \tag{9}$$

Thc first stage of this algorithm is the filling of the similarity matrix H. The matrix size is $m+1$ on $n+1$, where m and n are lengths of the corresponding sequences.

Matrix H is built in the following manner:

$$H(i,0) = 0,\ 0 \le i < m \qquad (10)$$

$$H(0,j) = 0,0 \le j < n$$

$$H(i,j) = max\begin{cases} 0 \\ H(i-1,j-1) + \omega(C_{ai},C_{bi}) \\ H(i-1,j) + \omega(C_{ai},-) \\ H(i,j-1) + \omega(-,C_{bi}) \end{cases} .$$

$$1 \le i \le m$$

$$1 \le j \le m$$

After the matrix is filed, the seeond stage of the algorithm is made. To obtain the optimum loeal alignment, the stage starts with the highest value in the matrix (i,j). This cell is marked as the current one. The next eurrent eell is the largest between the following: $(i\text{-}1,j)$, $(i,j\text{-}1)$, and $(i\text{-}1,j\text{-}1)$. In ease of equity between eells, priority is given to $h_{i,j}$. The proeess eontinues until it reaehes the cell with zero value, or the eell (0,0). After that the alignment is eonstrueted as next: starting with the last eurrent cell, the process reaches (i,j) using the previously-ealculated path. A diagonal jump implies an alignment (either a match or a mismatch). A top-down jump implies a deletion. A left-right jump implies an insertion.

The complexity of this algorithm is estimated as $O(m^*n)$.

3.2 Needleman-Wunsch Algorithm

The Needleman-Wunsch algorithm has few differences from local alignment algorithm. As in the Smith-Waterman algorithm, there are two sequenees on input: a = «$C_{a1},C_{a2}\ldots C_{an}$» and b = «$C_{b1},C_{b2}\ldots C_{bm}$», and the similarity function $S{:}(C_{a},C_{b},) \rightarrow Z$. One of the differenees from the Smith-Waterman algorithm is a constant d, which defines a penalty. The similarity function R is defined as:

$$R(a,b) = \sum_{i=0}^{n-1} f_i(a_i,b_i) \ . \qquad (11)$$

Where $f_i(a,b)$ is:

$$f_i(a,b) = \begin{cases} d,(a_i = -) \vee (b_i = -) \\ \omega(a_i,b_i),(a_i \neq -) \wedge (b_i \neq -) \end{cases} . \qquad (12)$$

This algorithm is also eonsists of two stages.

First of all, the similarity matrix S of size m+1 on n+1, where m and n are lengths of eorresponding sequences, is filled. The elements $s_{i,0}$ and $s_{0,j}$ are filled with values i^*d и j^*d, eorrespondingly. Other elements of S are ealeulated in the following manner:

$$F(i,j) = max\begin{cases} 0 \\ F(i-l,j-1)+\omega(C_{ai},C_{bi}) \\ F(i-1,j)+d \\ F(i,j-l)+d \end{cases} \quad (13)$$

$$1 \le i \le m$$
$$1 \le j \le m$$

On the second stage the current element is set to the bottom right. The next current element has to bc ehosen according to thc following conditions:

$$\begin{aligned} &s_{i-1,j-1}, s_{i,j} = s_{i-1,j-1} + \omega(C_{ai}, C_{bi}) \\ &s_{i-1,j}, s_{i,j} = s_{i-1,j} + d \\ &s_{i,j-1}, s_{i,j} = s_{i,j-1} + d \end{aligned} \quad (14)$$

In casc of meeting a couple or more conditions, the priority is given to the most top. The process is kept until it reaches the value in position (0,0).

The complexity of this algorithm is estimated as $O(m*n)$.

4 Intellectual Attack Detection

4.1 Detection of Attack Mutations

Let take a look on signature-based host-based IDS and presume that thc intruder's target is the attack implementation in way of evasion of IDS. Common ways of it are described in [5].

Tracc of system is dcfined by *SystemTrace* = «C_1, C_2, C_3...C_N». The *MaliciousTrace* = «C'_1, C'_2, C'_3...C'_M» is a trace corresponding to the attack. The problem of mutation detection is to discover in *SystemTrace* thc traces corresponding to the attaek mutation equal by a result to the attack implemented by *MaliciousTrace*.

The set $Seq = \{Seq_i, 0 \le i \le P, P \le N\}$ is built in the following way:

$$\begin{aligned} &Seq_1 = C_1, C_2 ... C_P \\ &Seq_2 = C_2, C_3 ... C_P \\ &... \\ &Seq_{N-P+1} = C_{N-P+1}, C_{N-P+2} ... C_N \end{aligned} \quad (15)$$

The elements of *Seq* are to bc compared with *MaliciousTrace*. Considering that $M << P$, the best algorithm for similarity calculating is the Smith-Waterman algorithm. As thc algorithm output there is a value *R*. In ease of its exceeding the valuc of a threshold it will bc considered that *SystemTrace* contains the mutation of attack implemented by *MaliciousTrace*. The choiee of threshold is thc main problem of this attack detection method.

The maximum number of commands in the mutated attack that this method can detect is P. But increase of P causes increase method working time. Each alignment is performed at $O(P*M)$. The quantity of alignment is $(N\text{-}P)$. So the complexity of method is estimated at:

$$O(P*M*(N\text{-}P)) = O(P*M*N\text{-}P^2*M)) . \qquad (16)$$

Thus P must be too big enough to detect a long attack and too small enough to satisfy the performance requirements. For the purpose of normalization it is suggested to use the next function instead of R:

$$R'(a,b) = \frac{R^2(b',b')}{R^2(b,b)} * \frac{Length(b)}{Length(b')} . \qquad (17)$$

Where b is *MaliciousTrace*, a is the elements of *Seq* set, b' is a *MaliciousTrace* sequence after alignment. In case of $R(a_l, a_l) \geq Length(a_l)$ for all sequences a_l, the definition range of this function matches the interval [0,1]. The nearness to null increases a probability of fact that this trace contains a mutation of attack implemented by *MaliciousTrace*.

Therefore, the sequences alignment algorithms can be used for attack mutation detection.

4.2 Anomaly Detection

The method of system calls sequence analysis is described in [6] and after that had some extensions in number of works, for example in [7], [8]. Let *SystemTrace* denotes a system trace corresponding to a normal behavior. The set $Seq = \{Seq_i, 0 \leq i \leq P, P \leq N\}$ is built in a following way:

$$\begin{aligned} Seq_1 &= C_1, C_2 ... C_P \\ Seq_2 &= C_2, C_3 ... C_P \\ &... \\ Seq_{N-P+1} &= C_{N-P+1}, C_{N-P+2} ... C_N \end{aligned} . \qquad (17)$$

NormDb is a set of elements *Seq*. This set represents a database of sequences that correspond to normal behavior.

Compare is a comparison function of two sequences:

$$Compare: Seq \times Seq \rightarrow \{0,1\} . \qquad (18)$$

The condition of addition the sequence Seq_k to a *NormDb*:

$$\begin{gathered} Seq_k \in NormDb \Leftrightarrow \forall Seq_i \in NormDb \wedge Seq_k \neq Seq_i : \\ Compare(Seq_k, Seq_i) = 0 . \end{gathered} \qquad (19)$$

The sequence will be added to database if it is not equal to any sequence in a database in terms of defined comparison function.

We suggest using the Needleman-Wunsch algorithm and *R'* function as a comparison function. The sequence is added to database in case of application of alignment algorithm to any this sequence and any sequence from a database the *R*' will be less than a threshold. The size of the database will be thus decreased.

5 The Results

For the testing on practice the UNIX system calls traces are used. The testing traces for *xlock*, *sendmail* и *lpr* programs are taken from University of New-Mexico site [9]. For each program there were a traces corresponding to normal behavior and attacks.

5.1 Detection of Attack Mutation

Firstly, it is necessary to define a ω function. Author of work [18] made an analysis how to divide the system calls in four groups from an IT security point of view. The first group is most dangerous, the fourth is the less. Therefore, the ω function can be defined in the following manner:

$$\omega(C_a,C_b)=\begin{cases} 3, C_a=C_b, C_a\in \mathrm{I} \\ 2, C_a=C_b, C_a\in \mathrm{II} \\ 1, C_a=C_b, C_a\in \mathrm{III} \\ 0, C_a=C_b, C_a\in \mathrm{IV} \\ -1, C_a\neq C_b \\ 0, (C_a=-)\vee(C_b=-) \end{cases}. \tag{19}$$

There were no attacks in original traces. The maximal values of *R* and *R'* were defined for each trace and for different values of *P*. Table 1 contains the test results.

Table 1. Results in case of normal traces without attacks

P	R	R'	time, sec
xlock trace of length 31729, buffer overflow attack 1			
100	7	0.08478	4,3
400	7	0.08478	8,9
700	10	0.00057	13,7
1000	10	0.00057	22,6
xlock trace of length 21182, buffer overflow attack 2			
100	7	0.07825	4,2
400	7	0.07825	8,8
700	7	0.07825	13,6
1000	7	0.07825	22,5
xlock trace of length 20973, buffer overflow attack 3			
100	11	0.12308	4,2
400	11	0.12308	8,8
700	11	0.12308	13,6
1000	11	0.12308	22,5
sendmail trace of length 32221, sunsendmailcp attack			
100	7	0.12088	1,3
400	9	0.01001	7
700	11	0.00932	14,2
1000	11	0.00932	21,6

The results meet the theory. As there are no attacks in the trace, *R'* is considerably less than 1. The method working time grows with *P* growing.

Then the mutated attacks were added to each trace. The mutated attack was obtained from common ones with adding 3, 6, 12, and 24 system calls that don't affect the attack goal (results are in table 2).

Table 2. Results in case of traces with attacks

Number of command added to common attack	R	R'
xlock trace of length 31729, buffer overflow attack 1		
3	17	0.8
6	17	0.667
12	17	0.5
24	17	0.333
xlock trace of length 21182, buffer overflow attack 2		
3	17	0.8
6	17	0.667
12	17	0.5
24	17	0.333
xlock trace of length 20973, buffer overflow attack 3		
3	17	0.8
6	17	0.667
12	17	0.5
24	17	0.333
sendmail trace of length 32221, sunsendmailcp attack		
3	16	0.8
6	16	0.667
12	16	0.5
24	16	0.333

It is seen from the results that R' values for normal traces and traces with attacks are considerably different. So this method can be used for the mutations detection.

5.2 Detection of Attack Mutations

The ω function was defined in a following way:

$$\omega(C_a,C_b)=\begin{cases} 3, C_a = C_b \\ -2, C_a \neq C_b \end{cases} . \qquad (20)$$

The penalty value *d* was set to -1. The threshold value was set to 0,7. The diagrams presented on figure 1 were built for different values of *P*.

The figure shows that the size of the database was considerably decreased. After that it was checked that decreasing has no affect on the detection ability of primary method. It is demonstrated in tables 3 and 4.

Estimating the performance, the primary method is better because time of common comparison is proportional to *P*, and sequences alignment is proportional to P^2. For the decreased size of the database, the suggested method takes 1,5-2 times less than primary.

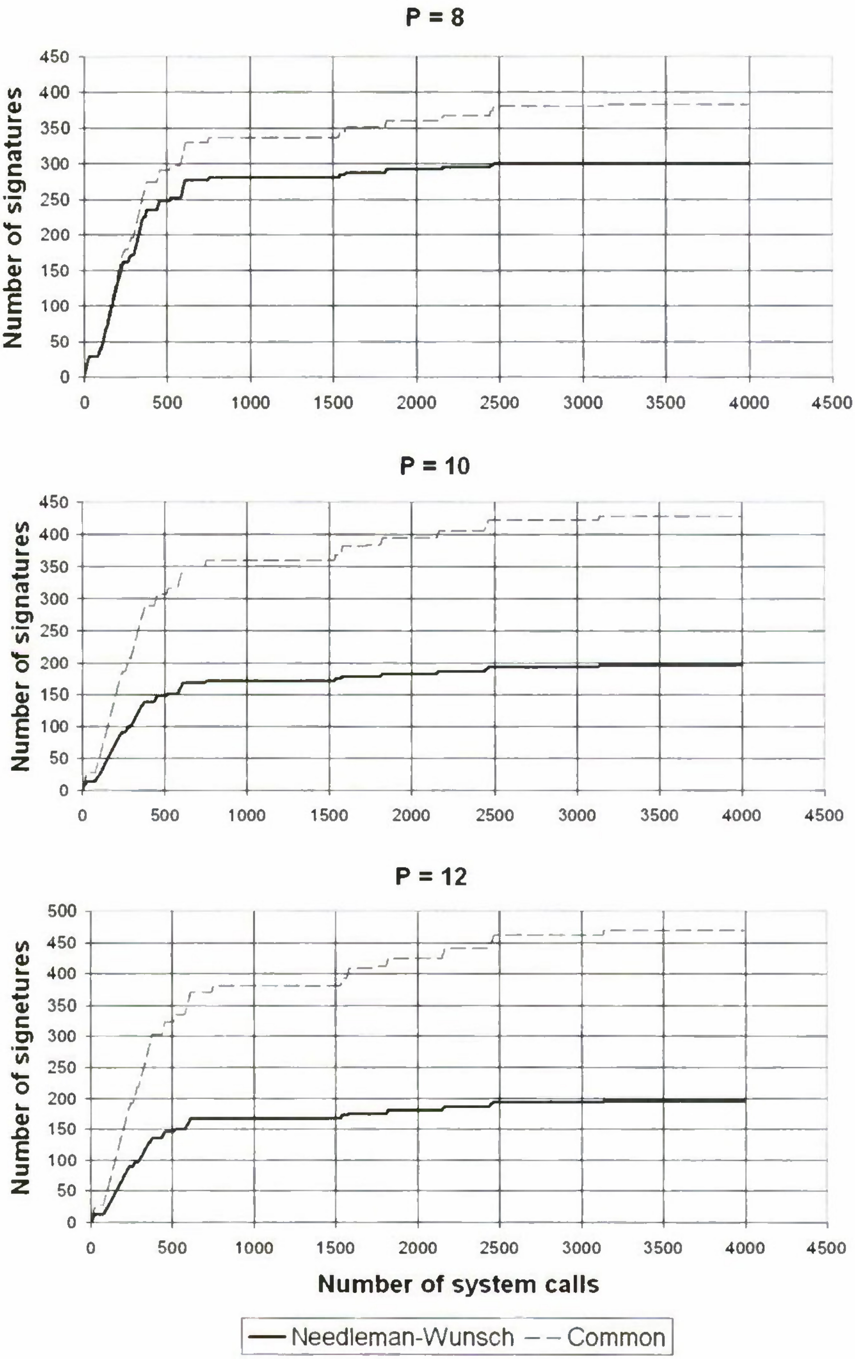


Fig. 1. Comparison of relations between database size and system calls number

Table 3. Comparison of anomaly percentage in case of traces without attacks

Anomaly percentage in normal trace	N = 8		N = 10		N = 12	
	N-W	Common	N-W	Common	N-W	Common
sendmail	0,30	0,32	0,40	0,84	0,44	0,52
xlock	0,11	0,11	0,03	0,10	0,03	0,10
lpc	0,38	0,41	0,41	0,56	0,35	0,57

Table 4. Comparison of anomaly percentage in case of traces with attacks

Anomaly percentage in trace with attack	N = 8		N = 10		N = 12	
	N-W	Common	N-W	Common	N-W	Common
sunsendmailcp	22,5	20,7	22,5	20,7	24,2	24,2
decode	23,5	23,5	23,5	19,9	27,2	26,1
syslog-local	30,7	30,7	30,7	28,6	32,3	31,4
syslog-remote	39,8	39,8	41,6	39,8	43,1	42,2
buffer overflow xlock 1	42,0	41,8	44,2	44,1	48,4	48,4
buffer overflow xlock 2	42,4	42,0	43,3	43,0	47,5	47,4
lprcp	32,4	32,1	34,5	34,4	37,6	37,1

The suggested method is thus an improvement of primary because it detects attacks with the same effectiveness and uses the decreased database.

6 The Related Works

There are a few works related to the sequences alignment algorithms used at malicious activity detection.

In [10] the sequences alignment algorithms are reviewed for the pattern matching. Approach was to detect a masquerade of normal user behavior by the intruder. Authors got some positive results in comparison to other algorithms Hybrid Markov and IPAM. In [11] the sequences alignment was suggested to generate the attack signatures for the purposes of detecting polymorphic attacks. The generation is focused on the string mode so it is considerably different from the method suggested in our work.

In [5], the method of attack mutation detection was proposed. They defined a set of no-ops calls. The suggested approach was consisted in searching any sequence that is equal to attack signature after deletion no-ops.

The approach suggested in our paper is more unified, flexible and effective than the analyzed techniques. For example, if there is a system with two different commands that implement the similar operation, our method can detect every kind of attack mutations which were obtained through the command replacing.

7 Conclusion

The paper reviews two sequence alignment algorithms. The results obtained for the first method showed that some parameter which is a criterion for attack detection is considerably different between normal traces and traces with attacks. It means that this method can be used in practice for mutated attack detection. It means that in IDS one signaturc can be used for detection of multiple attacks, even those attacks that are likely to be unknown. Also this method is helpful in reducing the size of the signature database in case of signature-based IDS. It is very important to reduce the database, because a number of attacks grows exponentially.

The future work has the objectives to investigate the use of sequences alignment algorithms to detcct the mutations in computer viruses.

The results obtained for the second algorithm have shown that comparing it to primary slide window method is more effective in memory and time usage. Ability of attack detection is not changing by using this method. It is also important that this method can be applied to any anomaly-based intrusion detection algorithm that uses any kinds of sequences to build behavior profile.

References

1. Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in Operating Systems. Communications of ACM 19(8), 461–471 (1976)
2. Needleman, S.B., Wunsch, C.D.: A gcneral method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48(3), 443–453 (1970)
3. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. Journal of Molecular Biology 147, 195–197 (1981)
4. Kruegel, C., Mutz, D., Valeur, F., Vigna, G.: On the Detection of Anomalous System Call Arguments. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 326–343. Springer, Heidelberg (2003)
5. Wagner, D., Soto, P.: Mimicry attacks on host-based intrusion detection systems. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 255–264. ACM Press, New York (2002)
6. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for unix processes. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy, pp. 120–128. IEEE Computer Society Press, Los Alamitos (1996)
7. Data Mining Approaches for Intrusion Detection, `http://www1.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html`
8. Leon, E., Nasraoui, O., Gomez, J.: Network Intrusion Detection Using Genetic Clustering. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 1312–1313. Springer, Heidelberg (2004)
9. Computer Immune Systems, `http://www.cs.unm.edu/~immsec/systemcalls.htm`
10. Coull, S.E., Branch, J.W., Szymanski, B.K., Breimer, E.: Intrusion Detection: A Bioinformatics Approach. In: Proceedings of 19th Annual Computer Security Applications Conference, pp. 24–33. IEEE Computer Society, Los Alamitos (2003)
11. Li, N., Xia, C., Yang, Y., Wang, H.: An Algorithm for Generation of Attack Signaturcs Based on Sequences Alignment. In: Proceedings of CSSE, vol. 3, pp. 964–969. IEEE Computer Society, Los Alamitos (2008)

Symptoms-Based Detection of Bot Processes

Jose Andre Morales[1], Erhan Kartaltepe[1], Shouhuai Xu[1,2], and Ravi Sandhu[1]

[1] Institute for Cyber Security, University of Texas at San Antonio
{jose.morales,erhan.kartaltepe,ravi.sandhu}@utsa.edu
[2] Department of Computer Science, University of Texas at San Antonio
shxu@cs.utsa.edu

Abstract. Botnets have become the most powerful tool for attackers to victimize countless users across cyberspace. Previous work on botnet detection has mainly focused on identifying infected bot computers or IP addresses and not on identifying bot processes on a host machine. This paper aims to fill this gap by presenting a bot process detection technique based on process symptoms such as: TCP connection attempts, DNS activities, digital signatures, unauthorized process tampering, and process hiding. We partition symptoms into sets which are input into classifiers generating individual detection models which are later appropriately integrated so as to improve the detection accuracy. The integrated approach correctly identified two bot processes and did not produced any false positives and false negatives.

Keywords: Botnet detection, bot process, process symptom, behavior-based detection, symptom-based detection.

1 Introduction

Botnets are an effective tool in spam distribution, denial of service attacks, illegal content hosting and other malicious acts. By leasing botnets, malware authors have successfully implemented profitable business models. These dynamic structures consist of several infected host machines (bots) running the bot software and responding to the bot master's instructions. Previous work on detection has mainly focused on the identification of infected bot machines or IP addresses, and not the actual bot process executing on the infected machine. This research presents three sets of process-based symptoms drawn from known bot samples — bot network activity behavior, unreliable provenance and stealth mechanisms — that are integrated together to detect bot processes on a host machine. Specifically, we make the following contributions:

- The process-based identification of (1) Bot network activity behavior: failed TCP connection attempts, DNS and reverse DNS queries; (2) Process provenance: using static file image digital signature verification and process/file system tampering; (3) Stealth mechanisms: using the absence of a graphical user interface and no required user input to execute.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 229–241, 2010.

- A formal detection model based on a non-trivial use of established data mining algorithms (C4.5). We conducted a thorough experiment on generating and evaluating detection models. Results show our methodology leads to better detection accuracy for both centralized and Peer-to-Peer (P2P) bots than a straightforward use of established data mining algorithms.

In both centralized and P2P structures, a bot must establish a connection to participate in the botnet possibly producing several failed connection attempts. Bots use DNS activity to reduce failed connection attempts which may instead produce failed DNS activity. In general, a process will attempt to connect to the input IP address of a successful reverse DNS query and the returned IP address of a successful DNS query, concluding the address is active. Our experiments reveal a counterintuitive approach that some bots attempt connecting to IP addresses regardless of DNS activity results: IP addresses that did not return a reverse DNS record are connected to successfully and IP addresses that did return a reverse DNS record failed to connect. Upon host infection, bot activity may manifest in one or more currently running processes. Bot processes may lack a digital signature, or may have been tampered with by a process lacking a digital signature. Bots typically execute without user knowledge by implementing stealth mechanisms, such as lacking a graphical user interface (GUI), not requiring keyboard and mouse input, removing itself from the list of currently active processes, and so on [16].

The rest of the paper is organized as follows: Section 2 is related work, Section 3 presents our bot detection methodology, Section 4 describes the chosen symptoms, Section 5 details the experimentation, results and limitations, and Section 6 gives our conclusion and future work.

2 Related Work

Network-based research analyzing botnets such as [7,2,12] use different techniques characterizing breadth and depth of centralized and P2P botnets, types of performed malicious activities, botnet structures, intrinsic events in the botnet life cycle and hiding techniques. Botnet detection research such as [5,6,1,4,8] primarily analyze network traffic using destination IP addresses, IRC server names, packet content, sequence of intrinsic bot events, crowd response and spatial-temporal relationships in their detection techniques. This results in the identification of several infected host machines as members of a centralized or P2P botnet. The research presented by Zhu et. al. [18] is a host-based detection technique of bots primarily based on a high rate of failed connection attempts. Connection failure rates of known bots are measured against benign processes and show that bots can be identified and distinguished from benign processes based on this single metric. Only measuring failed connection attempts may be most effective with IP addresses of dead botnets, discovered botnets and partially active botnets. However, a single metric is not enough to detect active bots which can possibly lead to the production of false negatives, especially with bots designed to limit failed connection attempts. Our research uses the novel approach of analyzing failed connection attempts in relation with DNS activity as the basis of our bot network activity behavior symptoms, along with symptoms for unreliable provenance and stealth mechanisms facilitating the identification of several metrics of suspicions processes producing a more robust detection technique. Establishing relationships between observed network data of a process is novel to this research as most related work considered observed network data in an isolated or sequential form. Relating together different observed network behaviors reveals dependencies bot processes have on various network services. Analyzing these dependencies facilitates deeper understanding of bot behavior which may not be appreciable in isolated or sequential analysis of observed network data. Our approach compliments these two forms of analysis and enhances understanding by adding a new perspective on bot behavior.

3 Bot Detection Methodology

Our model's premise is that bot and benign processes will exhibit different recognizable characteristics that can be utilized via appropriate algorithms. The differences may be characterized by a set of attributes mapped to a set of symptoms. Let us denote by A the universe of bot process attributes and by P a process currently executing on a host machine with symptoms $P_{symp_1} \ldots P_{symp_{|A|}}$ with respect to A. The goal is to determine the predicate $Bot(P)$, which determines if P is a bot, `true` means "yes" and `false` means "no". We want to identify a function $\mathtt{f}$ that computes $Bot(P) = \mathtt{f}(P_{symp_1} \ldots P_{symp_{|A|}})$. We can approximate the unknown $\mathtt{f}$ via a function f. A straightforward construction of f would be output of an established data mining algorithm, denoted as f_0. Such f_0 may not offer the desired detection accuracy, inspiring us to propose the following methodology: We can appropriately (1) partition the attributes A into multiple subsets based on certain domain knowledge, (2) generate a function g_i corresponding to the symptoms with respect to each partition of attributes, and (3) create function f based on composing the individual functions g_i.

Specifically, we propose to partition attributes based on the following "life-cycle" perspective of bot processes: A_1, *bot process network activity behavior*; A_2, *bot process provenance*; A_3, *bot process stealth mechanisms*. With respect to A_1, we hope to approximate the predicate $B(P)$, which indicates if P is exhibiting bot network activity behavior via a function g_1. With respect to A_2, we hope to approximate the predicate $U(P)$, which indicates if P has a unreliable provenance via a function g_2. With respect to A_3, we hope to approximate the predicate $S(P)$, which indicates if P has employed stealth mechanisms via specific known techniques via a function g_3. The desired function f can be constructed using g_1, g_2 and g_3 with flexible use of data mining techniques coupled with expert knowledge. We partition the bot process symptoms into three subsets where a symptom represents an occurred execution event or a property that is present during the life cycle of a bot process.

Approximating the predicate $B(P)$ with function $g_1(P)$. Intuitively, $g_1(P)$ analyzes network activity of a process P with a set of symptoms determining if P

is exhibiting similar network activity of known bots. The set of symptoms B_{sym} consist of $n \geq 1$ symptoms where each b_s describes a symptom of network activity previously observed in a known bot sample. P_{val} is a set of m responses θ from a process P which forms a one-to-one mapping with each b_s in B_{sym} and is used to determine if $B(P)$ is `true` or `false`. The values of θ_r are acquired by analyzing the network activity behavior of a process P during execution. The function $g_1(P)$ returns `true` if and only if there exists a value p_r with $\theta_r = true$ corresponding to a symptom b_s in B_{sym}, thus we have:

$$\begin{aligned} &B_{sym} = \{b_1 \ldots b_s \ldots b_n\}, P_{val} = \{p_1 : \theta_1 \ldots p_r : \theta_r \ldots p_m : \theta_m\} \\ &B(P) = g_1(P) = \texttt{true} \Leftrightarrow \exists b_s, p_r : (s = r \wedge \theta_r = \texttt{true}) \end{aligned} \quad (1)$$

When $\theta_r =$ `true`, P exhibited the described network activity of symptom b_s. If all θ_r evaluate to `false`, then P does not exhibit bot behavior and $g_1(P) =$ `false`; but if just one θ_r evaluates to `true` then P has the specific symptom b_s and $g_1(P) =$ `true`.

Approximating the predicate $U(P)$ with function $g_2(P)$. Intuitively, $g_2(P)$ compares origin information of a given process P with a set of symptoms $u_s \in U_{sym}$ deciding if the process's provenance is reliable. This predicate asks the question: has the origin of process P been malevolently tampered or created making it unreliable? A response of `true` indicates it is not reliable; `false` indicates it is. The symptoms are a list $u_1 \ldots u_s \ldots u_n$, $n \geq 1$ submitted to a process P which returns a set of values P_{val}, $p_1 \ldots p_r \ldots p_m$, and compared with U_{sym}. Each symptom u_s precisely states a singular scenario of process unreliability previously observed in a known bot sample which tampered or created another process in a malevolent manner. Each result $p_r \in P_{val}$ contains an answer $\xi_r =$ `true` or `false`, which corresponds to the claim $u_s \in U_{sym}$. The function $g_2(P)$ will return `true` if and only if an answer ξ_r of a result $p_r \in P_{val}$ is `true`, thus we have:

$$\begin{aligned} &U_{sym} = \{u_1 \ldots u_s \ldots u_n\}, P_{val} = \{p_1 : \xi_1 \ldots p_r : \xi_r \ldots p_m : \xi_m\} \\ &U(P) = g_2(P) = \texttt{true} \Leftrightarrow \exists u_s, p_r : (s = r \wedge \xi_r = \texttt{true}) \end{aligned} \quad (2)$$

If P_{val} returns all `false` answers then it is reliable and $g_2(P) =$ `false`; if just one $p_r \in P_{val}$ has a value $\xi_r =$ `true`, then P's provenance is not reliable and $g_2(P) =$ `true`.

Approximating the predicate $S(P)$ with function $g_3(P)$. Intuitively, $g_3(P)$ determines if a process P is implementing stealth mechanisms previously observed in a known bot sample. The set of symptoms S_{sym} consist of $n \geq 1$ symptoms $s_1 \ldots s_s \ldots s_n$ where each s_s describes a specific stealth mechanism previously observed in a known bot sample. P_{val} is a set of m responses μ_z from a process P which forms a one-to-one mapping with each S_s in S_{sym} and is used to determine if $S(P)$ is `true` or `false`. μ_r values are acquired by analyzing the execution behavior of process P for the possible use of known stealth mechanisms. The function $g_3(P)$

returns true if and only if there exists a value p_r with μ_r = true corresponding to a symptom $s_s \in S_{sym}$, thus we have:

$$S_{sym} = \{s_1 \ldots s_s \ldots s_n\}, P_{val} = \{p_1 : \mu_1 \ldots p_r : \mu_r \ldots p_m : \mu_m\}$$
$$S(P) = g_3(P) = \texttt{true} \Leftrightarrow \exists s_s, p_r : (s = r \wedge \mu_r = \texttt{true}) \quad (3)$$

The implication of μ_r = true is that P exhibited the specific known stealth mechanism described in symptom s_s. If all μ_r evaluate to false, then P does not exhibit known stealth mechanisms and $g_3(P)$ = false; but if just one μ_r evaluates to true then P has the specific symptom s_s and $g_3(P)$ = true.

Approximating the predicate f with function f based on functions g_1, g_2 and g_3. We approximate f via a function f by utilizing g_1, g_2 and g_3. Three example definitions to determine $Bot(P)$ are:

$f_1(P) = g_1(P) \vee (g_2(P) \wedge g_3(P))$. This is the least restrictive, since a process is deemed a bot when it exhibits bot network activity behavior or has both an unreliable provenance and has stealth mechanisms. False positives can be produced by benign processes with an instance of bot network activity behavior such as a process with a successful connection attempt to the input IP address of a failed reverse DNS query.

$f_2(P) = g_1(P) \wedge (g_2(P) \vee g_3(P))$. This is more restrictive, the and ($\wedge$) operator requires a process to exhibit bot behavior and either unreliable provenance or stealth mechanisms. This will focus detection more on processes with bot-like behavior. False positives can arise with benign processes lacking a digital signature, thereby giving them unreliable provenance, while having an instance of bot network activity behavior such as a failed connection attempt to the input IP address of a successful reverse DNS query. This definition excludes possible detection of bots that possess unreliable provenance and/or stealth mechanisms but do not show bot network activity behavior.

$f_3(P) = g_1(P) \wedge g_2(P) \wedge g_3(P)$. This is the most restrictive with the and ($\wedge$) operators requiring triple analysis with each component returning true. A process is deemed a bot when it exhibits bot network activity behavior, unreliable provenance and stealth mechanisms. This detection has the highest probability of identifying malicious bots, and excluding benign processes. A process with an unreliable provenance or exhibiting bot behavior or stealth mechanisms is assumed benign which could produce a false negative.

4 Symptoms of Bot Processes

Bot Behavior Symptoms. Evaluating a process's bot network activity behavior employed three symptoms in set B_{sym}. All the symptoms were based on a process P, on a local machine, interacting with and responding to TCP protocol connection attempts and DNS activity (DNS queries and reverse DNS queries).

b_1: Failed connection attempt to the returned IP address of a successful DNS query. This is considered abnormal behavior; a successful DNS query suggests the returned IP address is active and can establish connections. Many of these IP addresses failing to connect were also the input IP of a failed reverse DNS query.

b_2: **IP address in a successful DNS activity and connection.** This is considered normal behavior. A DNS activity can be either a DNS query or a reverse DNS query. More precisely, we consider the returned IP address of a successful DNS query or the input IP address of a successful reverse DNS query which is also used in a successful connection. In our analysis several more bots than benign processes connected to such IP addresses. This further implies the dependency bots have on DNS activity when attempting connections to remote hosts.

b_3: **Connection attempt to the input IP address of a failed reverse DNS query.** This is considered abnormal behavior; an IP address failing a reverse DNS query should be presumed inactive and should not be used in a connection attempt. Almost all analyzed bot samples performed reverse DNS queries possibly to harvest new domain names of malware servers or infected hosts. Some bots failed to connect with the input IP addresses of a successful reverse DNS query and other bots successfully connected to input IP addresses of a failed reverse DNS query. This counterintuitive use of input IP addresses used in failed reverse DNS requests implies bots attempt TCP connections with IP addresses regardless of DNS activity results for reasons other than TCP connection attempts. One possible motivation may be the reverse DNS query is used solely to dynamically acquire during execution new IP addresses or domain names of malware servers, redirection servers or newly infected victim machines. This helps bots by having to store fewer IP address/domain name pairs in their static file images prior to initial execution; thereby making it harder for security personnel to predetermine the structure and components of a botnet just through static file image analysis.

Unreliable Provenance Symptoms. Determining the provenance of a process employed three symptoms in set U_{sym}. Selection of these symptoms were based on verifying the existence of a static file image's digital signature for known bot and benign files, files of bot's parent process, and analyzing process memory for unauthorized modification by some other process primarily through dynamic code injection. The absence of a digital signature in a file or the parent file that created it raises suspicion due to its unknown origin. All of our bots and a few benign static file images lacked a digital signature. Most of the static file images of benign software installers were digitally signed. Dynamic code injection, mostly a malevolent technique coercing a process into unauthorized behavior [16,15], is frequently used by our analyzed bots on benign processes which then exhibit bot behavior.

u_1: **Standalone executable's static file image does not have a digital signature.** A standalone executable file is written directly to the file system without an installer. Malware contained in email attachments, website downloads and portable memory infect a system this way [16]. The majority of the analyzed benign standalone executables had digital signatures. All of our standalone bot samples lacked a digital signature.

u_2: **Dynamic code injector's static file image does not have a digital signature.** Dynamic code injection is used by bots to infect legitimate processes

[16,3]. There are benevolent uses for this, such as debugging and detection of suspicious activity where the injector's static file image almost always contains a digital signature. Bot injectors typically do not have digital signatures. A process whose injector lacks a digital signature is identified as having an unreliable provenance since the injector's origin cannot be established.

u_3: **Creator of process's static file image does not have a digital signature.** Bots will self-replicate or install other malware on a system [16]. The newly created malware may or may not have a digital signature but the malware installer will likely lack a digital signature, which is considered unreliable provenance. An installed file lacking a digital signature with its installer having a digital signature is considered to have a reliable provenance.

Stealth Mechanism Symptoms. Evaluating a process's stealth mechanisms employed two symptoms in set S_{sym}, all based on a process P's use of graphical user interfaces (GUI) along with reading keyboard and mouse inputs.

s_1: **Graphical user interface.** A vast amount of benign software interact with the user via a GUI. Bots typically do not use a GUI since it calls attention to their existence and may result in their termination [16]. A process executing without a GUI is considered to have a stealth mechanism.

s_2: **Human computer interface.** A benign program may require user input to execute an operation; this is typical interaction between application and user. Bots tend to execute their nefarious acts without the need of explicit user input. A process executing without reading keyboard or mouse events is considered to have a stealth mechanism.

5 Experiment and Results

Data Collection and Instrumentation. Bot data collection was done using VMWare Workstation running Microsoft Windows XP SP2 with no updates and no antivirus. Four active bots: `virut`, `waledac`, `wopla`, `bobax`, and five inactive bots: `nugache`, `wootbot`, `gobot`, `spybot`, `storm`, were executed for a twelve hour period. These centralized and P2P bots possess different stealth mechanisms, diverse command and control channels, various packet encryption and self updates. Packets were captured using `Windows Network Monitor`. Detecting dynamic code injection and Bot replication was accomplished with a real time monitor implementing known techniques [15,11]. Digital signature verification of static file images was done using `Sigcheck` [14]. An enhanced version of `GlobalHook` [10] was used to collect keyboard and mouse input, GUI presence was recorded using `EasyHook` [9]. Collecting data of known benign processes was performed on two verified malware-free desktops running Windows XP SP2 for twelve hours during which both machines performed several network-based activities including web browsing, FTP, instant messaging, P2P file sharing and software updates. The collection, with 20 bot processes and 62 benign processes (41 different applications with some being tested multiple times) listed in Table 1, produced a diversity of symptom combinations. In Table 1, most of the symptoms have {Yes,No} values

Table 1. Bot and Benign Processes Used in the Training Set

Bot Processes									
Bot	Process	Bot Network			Unreliable			Stealth	
Name	Name	Activity Behavior			Provenance			Behavior	
		b_1	b_2	b_3	u_1	u_2	u_3	s_1	s_2
Nugache	mstc.exe	Yes	0	Yes	No	Yes	Yes	No	No
Virut	Svchost.exe	Yes	0	Yes	No	Yes	No	No	No
	Svchost.exe	Yes	0	Yes	No	Yes	No	No	No
	winlogon.exe	No	2	Yes	No	Yes	No	No	No
	svchost.exe	No	1	Yes	No	Yes	No	No	No
	svchost.exe	No	0	Yes	No	Yes	No	No	No
	svchost.exe	No	0	Yes	No	Yes	No	No	No
	svchost.exe	No	2	Yes	No	Yes	No	No	No
Waledac	Save.exe	Yes	123	Yes	Yes	No	No	No	No
Wopla	Rundll32.exe	Yes	1	Yes	No	Yes	No	No	No
Bobax	Explorer.exe	No	2	No	No	Yes	No	No	No
Wootbot	videosd32.exe	Yes	0	No	No	Yes	Yes	No	No
Gobot	Gobot-o.exe	Yes	0	Yes	Yes	No	No	No	No
Spybot	wuaghqr.exe	Yes	0	No	No	Yes	Yes	No	No
Storm	testdll_f.dll	Yes	0	Yes	Yes	No	No	No	No
Bobax	Explorer.exe	Yes	2	Yes	No	Yes	No	No	No
Wopla	Rundll32.exe	No	4	Yes	No	Yes	No	No	No
Waledac	waledac.exe	Yes	7	Yes	Yes	No	No	No	No
Virut	winlogon.exe	No	2	Yes	No	Yes	No	No	No
	svchost.exe	No	2	Yes	No	Yes	No	No	No
Benign Processes									
360tray		Flock			Mercury			Skype	
AOL Explorer		Foxmail			MS Messenger			Snarfer	
Avant		Google Chrome			Msfeedssync			stormliv	
Bittorrent		googlepinyIndaemon			Mstc			Svchost	
BlogBridge		Internet Explorer			Opera			ThinReader	
Btdna		Jusched			Ppstream			ThunderBird	
ccApp		Kaspersky AV			RSS Bandit			WinSCP3	
Cuteftp32		K-Meleon			RSS Owl			wlcomm	
Explorer		LimeWire			Rundll32			wlmail	
FeedReader		Maxthon			SeaMonkey			Xdict	
Firefox									

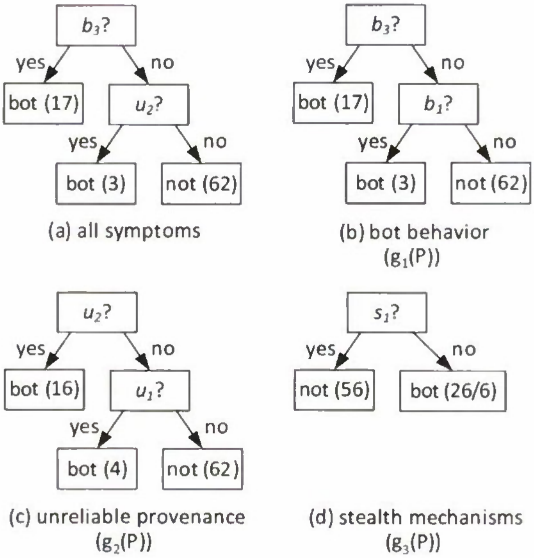
with Yes ↦ `true` and No ↦ `false`, except in s_1 and s_2 where Yes ↦ `false` and No ↦ `true`. Symptom b_2 is considered normal behavior and presented as a total occurrence amount. Test data was collected using five laptops, with minimal security and no recent malware scans, for eight to twelve hours. A post-test data collection malware scan of all five laptops revealed two bot processes: `servwin.exe` as the *cutwail bot*, which was not part of the training set, and `TMP94.tmp` as the *Virut bot*. The test set, listed in Table 2 consisted of 34 processes including two bot

Table 2. Test Set: Decision Tree and Bot Process Predictions

Process	Bot Network				Unreliable				Stealth			Bot			
Name	Activity Behavior				Provenance				Behavior			Prediction			
	b_1	b_2	b_3	$B(P)$	u_1	u_2	u_3	$U(P)$	s_1	s_2	$S(P)$	f_0	f_1	f_2	f_3
svchost.exe	N	0	N	F	N	N	N	F	N	N	T	F	F	F	F
googletalk.exe	N	2	N	F	N	N	N	F	Y	Y	F	F	F	F	F
firefox.exe	N	5	N	F	N	N	N	F	Y	Y	F	F	F	F	F
cutftp32.exe	Y	1	N	T	Y	N	N	T	N	N	F	F	T	T	F
firefox.exe	N	44	N	F	N	N	N	F	Y	Y	F	F	F	F	F
svchost.exe	N	0	N	F	N	N	N	F	N	N	T	F	F	F	F
servwin.exe	Y	0	Y	T	Y	N	N	T	N	N	T	T	T	T	T
Framework Services.exe	N	1	N	F	N	N	N	F	N	N	T	F	F	F	F
iexplore.exe	N	126	N	F	N	Y	N	T	Y	Y	F	T	F	F	F
firefox.exe	N	49	N	F	N	Y	N	T	Y	Y	F	T	F	F	F
rundll32.exe	N	1	N	F	N	N	N	F	N	N	T	F	F	F	F
firefox.exe	N	67	N	F	N	N	N	F	Y	Y	F	F	F	F	F
firefox.exe	N	7	N	F	N	N	N	F	Y	Y	F	F	F	F	F
iexplore.exe	N	54	N	F	N	N	N	F	Y	Y	F	F	F	F	F
firefox.exe	N	45	N	F	N	N	N	F	Y	Y	F	F	F	F	F
firefox.exe	N	10	N	F	N	N	N	F	Y	Y	F	F	F	F	F
SshClient.exe	N	1	N	F	Y	N	N	T	Y	Y	F	F	F	F	F
BitLord.exe	Y	1	N	T	Y	N	N	T	N	N	F	F	T	T	F
Acrobat.exe	N	1	N	F	N	N	N	F	Y	Y	F	F	F	F	F
Thunder5.exe	Y	13	N	T	N	N	N	F	Y	Y	F	F	T	F	F
Thunder Minisite.exe	N	7	N	F	N	N	N	F	Y	Y	F	F	F	F	F
Thunder5.exe	Y	24	N	T	N	N	N	F	Y	Y	F	F	T	F	F
wmplayer.exe	Y	17	N	T	N	N	N	F	Y	Y	F	F	T	F	F
setup_wm.exe	N	1	N	F	N	N	N	F	Y	Y	F	F	F	F	F
chrome.exe	N	3	N	F	N	N	N	F	Y	Y	F	F	F	F	F
TMP94.tmp	N	3	Y	T	N	Y	N	T	N	N	T	T	T	T	T
Google Update.exe	N	1	N	F	N	N	N	F	N	N	T	F	F	F	F
Google Update.exe	N	1	N	F	N	N	N	F	N	N	T	F	F	F	F
chrome.exe	N	28	N	F	N	N	N	F	Y	Y	F	F	F	F	F
Adobe_ Updater.exe	N	2	N	F	N	N	N	F	Y	Y	F	F F	F F	F F	F F
gup.exe	N	1	N	F	N	N	N	F	Y	Y	F	F	F	F	F
Tvanst.exe	Y	1	N	T	Y	N	N	T	N	N	F	F	T	T	F
msfeeds															
sync.exe	N	I	N	F	N	N	N	F	N	N	T	F	F	F	F
zclientm.exe	N	I	N	F	N	N	N	F	N	N	T	F	F	F	F

processes, the rest were assumed benign. One of the bot processes and several of the benign in the test set were not part of the training set.

J48 classification decision trees. The results presented here are partially based on the J48 decision trees [17] in Figure 1. Running the training sets containing bot behavior, unreliable provenance and stealth mechanisms individually produced the decision trees in Figures 1(b), 1(c) and 1(d). Each leaf node, shown as a rectangle, represents the total number of processes classified as exhibiting (`=yes`) or not exhibiting (`=no`) the symptom of the leaf node's parent. A summation of the numeric values in appropriate leaf nodes gives the total number of processes with a (`=yes`) or (`=no`) answer. The bot network activity behavior decision tree in Figure 1(b) produced eight `true` responses with the test set data. The two bot processes were amongst the eight; six false positives and no false negatives were produced. The unreliable provenance decision tree in Figure 1(c) produced eight `true` responses with the test data. The two bot processes were amongst the eight; six false positives and no false negatives were produced. Five processes exhibited unreliable provenance symptom u_1 and three processes exhibited unreliable provenance symptom u_2. Two of the three processes with symptom u_2 were purposely injected (see paragraph below **The cases of** f_0,f_1,f_2 **&** f_3). The stealth mechanisms decision tree in Figure 1(d) produced ten `true` responses with the test data. The two bot processes were amongst the ten; eight false positives and no false negatives were produced. The high amount is a result of having many system and software update processes in the test set that are known to run without a GUI. The two bot processes had no GUI which is assumed implemented as part of a larger stealth strategy [16].



(a) all symptoms

(b) bot behavior (g_1(P))

(c) unreliable provenance (g_2(P))

(d) stealth mechanisms (g_3(P))

Fig. 1. J48 Decision Trees Used in f_0, f_1, f_2 and f_3

The cases of f_0,f_1,f_2 & f_3. Evaluating the test data results of bot behavior $B(P)$, unreliable provenance $U(P)$ and stealth mechanisms $S(P)$ with f_0, f_1, f_2, and f_3 are listed in Table 2 along with final bot predictions. The case of f_0 is a simplistic use of the J48 classifier. Using all the symptoms to analyze the training set data produced the decision tree in Figure 1(a) with no false positives and no false negatives. Analyzing the test set data with this decision tree produced two false positives and no false negatives, listed in Table 2. `RemoteDLL` [13] is a benevolent utility which lacks a digital signature that loads and removes DLLs from a process. In our test set, processes `iexplore.exe` and `firefox.exe` were purposely DLL injected using RemoteDLL producing two false positives with the decision tree in Figure 1(a) since the injector had no digital signature. In the case of f_1, our test set produced eight `true` responses including the two known bots, leaving six false positives and no false negatives. All eight exhibited bot behavior including the two bots which were also the only ones exhibiting unreliable provenance and stealth mechanisms. Bot behavior was highly prevalent, partly due to benign network active processes executing combinations of DNS activity with connection attempts. In the case of f_2, our test set produced five `true` responses including the two known bots, with three false positives, an improvement over f_1. All five exhibited unreliable provenance but only the two bots exhibited stealth mechanisms as well. Only the bots possessed additional symptoms, hinting more accurate performance can be made with stronger restrictions. In the case of f_3, our test set produced only two `true` responses: our two discovered bot processes. Perfect results were yielded by f_3 suggesting accurate detection with minimal false positives and false negatives may be achieved with high restriction enforcement.

Discussion. Only five of the eight symptoms, b_1, b_3, u_1, u_2, s_1, composed the decision trees and were used in the final bot predictions. Symptom s_1 was the most dominant with thirteen processes in our test set executing without GUI. This is not surprising as several tested benign processes were system services running in the background. Symptom b_1 occurred often due to processes failing to connect with the returned IP address of a successful DNS query. Symptom b_3 only occurred in the two test set bots, suggesting that a well designed benign application will not attempt to connect to IP addresses involved in a failed reverse DNS query while bots attempt connections regardless of DNS activity results. According to Table 2, only the bot processes `servwin.exe` (*cutwail bot*) and `TMP94.tmp` (*Virut bot*) possessed more than one bot behavior symptom. This hints to strong dependencies on DNS activities by bots and higher probability to attempt connections with IP addresses involved in DNS activities. Symptom u_1 occurred often due to our test set processes lacking digital signatures. One can assume that a portion of benign applications and the vast majority of malware will lack a digital signature. Both symptoms s_1 and s_2 precisely matched for each process in the test set, meaning every process executing with a GUI also read user input and every process without a GUI did not read any user input. Ranking from least effective to most effective detection produces: f_1, f_2, f_0, f_3. Even though f_0 was second most effective in bot detection, given a more diverse test set the straightforward construction of f_0 may not be so effective, as shown by our purposeful injection of two processes during testing. Detecting the most devious of bots may be best achieved with f_3 but f_2 may capture a broader range of bots possessing less symptoms. A combination of the restrictions of f_2 and f_3 may be best suited for bot detection and combining restrictions of f_1 may be the best to detect other non-bot malware.

Limitations. IP addresses of DNS activity not used in a connection attempt by a captured process were not analyzed since we could not reliably map specific DNS activity with a specific process. Only Win32 processes were analyzed while kernel processes were not. We are currently developing utilities eliminating these limitations allowing their inclusion in our evaluations.

6 Conclusion and Future Work

We presented in this research a symptoms-based technique for detecting bot processes using three distinct user defined sets of symptoms drawn from known bot samples: bot network activity behavior, unreliable provenance and stealth mechanisms. Through a non-trivial use of J48 classifier, three distinct evaluations were performed correctly identifying two bot processes. Bot network activity behavior symptoms were based on failed connection attempts and DNS activity; provenance symptoms were based on the existence of digital signatures and process/file system tampering; stealth mechanisms were based on the absence of a GUI and no required reading of user input. Several of the chosen symptoms appeared in both benign and bot processes, but the bot processes showed a much higher quantity and diversity of symptoms. Based on the results, the strongest restrictive analysis requiring symptoms of all three sets was the best singular detection solution producing no false positives and no false negatives. In dealing with future bots and other non-bot malware combining stronger and weaker restrictions may be a desirable detection approach. Future work includes analyzing kernel mode bots and a diverse set of network protocols, as well as a kernel-based real-time monitor detecting presence of bot processes.

Acknowledgments

This work is partially supported by grants from AFOSR, ONR, AFOSR MURI, and the State of Texas Emerging Technology Fund.

References

1. Collins, M.P., Shimeall, T.J., Faber, S., Janies, J., Weaver, R., De Shon, M., Kadane, J.: Using uncleanliness to predict future botnet addresses. In: IMC 2007: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, pp. 93–104. ACM, New York (2007)
2. Dagon, D., Gu, G., Lee, C.P., Lee, W.: A taxonomy of botnet structures. In: Computer Security Applications Conference, Annual, pp. 325–339 (2007)
3. Filiol, E.: Computer Viruses: from Theory to Applications. IRIS International series. Springer, Heidelberg (2005), iSBN 2-287-23939-1

4. Goebel, J., Holz, T.: Rishi: identify bot contaminated hosts by irc nickname evaluation. In: HotBots 2007: Proceedings of the First Conference on First Workshop on Hot Topics in Understanding Botnets, p. 8. USENIX Association, Berkeley (2007)
5. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Proceedings of the 17th USENIX Security Symposium, Security 2008 (2008)
6. Gu, G., Zhang, J., Lee, W.: BotSniffer: Detecting botnet command and control channels in network traffic. In: Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS 2008) (February 2008)
7. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.: Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In: LEET 2008: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, pp. 1–9. USENIX Association, Berkeley (2008)
8. Hu, X., Knysz, M., Shin, K.G.: Rb-seeker: Auto-detection of redirection botnets. In: 16th Annual Network and Distributed System Security Symposium (2009)
9. Husse, C.: Easyhook 2.6, `http://www.codeplex.com/easyhook`
10. Mamaladze, G.: Globalhook,
`http://www.codeproject.com/KB/cs/globalhook.aspx`
11. Morales, J.A., Clarke, P.J., Deng, Y., Kibria, B.G.: Identification of file infecting viruses through detection of self-reference replication. Journal in Computer Virology Special EICAR Conference Invited Paper Issue (2008)
12. Nazario, J., Holz, T.: As the net churns: Fast-flux botnet observations. In: 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, pp. 24–31 (2008)
13. Remote dll injection application,
`http://www.novell.com/coolsolutions/tools/17354.html`
14. Sigcheck 1.6,
`http://technet.microsoft.com/en-us/sysinternals/bb897441.aspx`
15. Sun, H.M., Tseng, Y.T., Lin, Y.H., Chiang, T.J.: Detecting the code injection by hooking system calls in windows kernel mode. In: 2006 International Computer Symposium, ICS 2006 (2006)
16. Szor, P.: The Art of Computer Virus Research and Defense. Symantec Press & Addison-Wesley (2005)
17. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
18. Zhu, Z., Yegneswaran, V., Chen, Y.: Using failure information analysis to detect enterprise zombies. In: 5th International ICST Conference on Security and Privacy in Communication Networks, Securecomm 2009 (2009)

A Comparison of Feature-Selection Methods for Intrusion Detection

Hai Thanh Nguyen, Slobodan Petrović, and Katrin Franke

Norwegian Information Security Laboratory
Gjøvik University College, Norway
{hai.nguyen,slobodan.petrovic,katrin.franke}@hig.no

Abstract. Feature selection is an important pre-processing step in intrusion detection. Achieving reduction of the number of relevant traffic features without negative effect on classification accuracy is a goal that greatly improves overall effectiveness of an intrusion detection system. A major challenge is to choose appropriate feature-selection methods that can precisely determine the relevance of features to the intrusion detection task and the redundancy between features. Two new feature selection measures suitable for the intrusion detection task have been proposed recently [11,12]: the correlation-feature-selection (CFS) measure and the minimal-redundancy-maximal-relevance (mRMR) measure. In this paper, we validate these feature selection measures by comparing them with various previously known automatic feature-selection algorithms for intrusion detection. The feature-selection algorithms involved in this comparison are the previously known SVM-wrapper, Markov-blanket and Classification & Regression Trees (CART) algorithms as well as the recently proposed generic-feature-selection ($GeFS$) method with 2 instances applicable in intrusion detection: the correlation-feature-selection ($GeFS_{CFS}$) and the minimal-redundancy-maximal-relevance ($GeFS_{mRMR}$) measures. Experimental results obtained over the KDD CUP'99 data set show that the generic-feature-selection ($GeFS$) method for intrusion detection outperforms the existing approaches by removing more than 30% of redundant features from the original data set, while keeping or yielding an even better classification accuracy.

Keywords: intrusion detection; feature selection; polynomial mixed $0-1$ fractional programming; mixed $0-1$ integer linear programming.

1 Introduction

The problem of intrusion detection is often analyzed as a pattern recognition problem - an Intrusion Detection System (IDS) has to tell normal from abnormal behaviour of network traffic and/or command sequences on a host. In addition, it is of interest to further classify abnormal behaviour in order to undertake adequate counter-measures. An IDS can be modeled in various ways (see for example [9], [10]). A model of this kind usually includes a representation algorithm

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 242–255, 2010.

(for representing incoming data in the space of selected features) and a classification algorithm (for mapping the feature vector representation of the incoming data to elements of a certain set of values, e.g. normal or abnormal etc.) Some IDS, like the ones presented in [9], also include the feature selection algorithm, which determines the features to be used by the representation algorithm. Even if the feature-selection algorithm is not included in the model directly, it is always assumed that such an algorithm is run before the very intrusion detection process.

The quality of the feature selection algorithm is one of the most important factors that affect the effectiveness of an IDS. The goal of the algorithm is to determine the most relevant features of the incoming traffic, whose monitoring would ensure reliable detection of abnormal behaviour. Since the effectiveness of the classification algorithm heavily depends on the number of features, it is necessary to minimize the cardinality of the set of selected features, without dropping potential indicators of abnormal behaviour. Obviously, determining a good set of features is not an easy task. The most of the work in practice is still done manually and the feature selection algorithm depends too much on expert knowledge. Automatic feature selection for intrusion detection is therefore important. For automatic feature selection, the wrapper and the filter models from machine learning are frequently applied [18]. The wrapper model assesses the selected features by learning algorithm's performance. Therefore, the wrapper method requires a lot of time and computational resources to find the best feature subsets. The filter model considers statistical characteristics of a data set directly without involving any learning algorithm. Due to the computational efficiency, the filter method is usually used to select features from high-dimensional data sets, such as intrusion detection systems. The filter model encompasses two groups of methods: the feature ranking methods and the feature-subset-evaluating methods. The feature ranking methods assign weights to features individually based on their relevance to the target concept. The feature-subset-evaluating methods estimate feature subsets not only by their relevance, but also by the relationships between features that make certain features redundant. It is well known that the redundant features can reduce the performance of a pattern recognition system. Therefore, the feature-subset-evaluating methods are more suitable for selecting features for intrusion detection. A major challenge in the IDS feature selection process is to choose appropriate measures that can precisely determine the relevance of features to the intrusion detection task and the relationship between features of a given data set.

Since the relevance and the relationship are usually characterized in terms of correlation or mutual information [4,19], we focus on two feature selection measures for intrusion detection task: the correlation-feature-selection (CFS) measure [1] and the minimal-redundancy-maximal-relevance (mRMR) measure [2]. In [11,12], a new search method that ensures globally optimal feature sets by means of the CFS and the mRMR measures was proposed. It was shown that the proposed search method outperforms the heuristic search strategies by removing much more redundant features from the KDD CUP 1999 data set [7] and still keeping the classification accuracies or even getting better performances. In this paper, the feature selection measures proposed in [11,12] are validated by comparison with various previously known automatic feature-selection algorithms for intrusion detection. Thus, the feature-selection algorithms involved in the comparison are the previously known SVM-wrapper [13], Markov-blanket [14] and CART [14] and the new generic-feature-selection ($GeFS$) method with 2 instances applied in intrusion detection: the correlation-feature-selection ($GeFS_{CFS}$) [11] and the minimal-redundancy-maximal-relevance ($GeFS_{mRMR}$) [12] measures.

A theoretical basis for comparison of the methods proposed in [11,12] and the other methods is difficult to give. Such a basis would require the general solution of the problem of comparison of filter and wrapper methods, which is not known (sometimes, the filter methods perform better, but sometimes the wrapper methods perform better). Because of that, in this paper we present the results of practical comparison achieved on a particular data set. Then the generalization of the results of the comparison depends to a large extent on the quality and generality of the test data set. We believe that the data set used for this comparison with the modifications deseribed below is general enough to elaim that our eomparison results can be generalized with high probability.

Any feature selection algorithm seleets relevant traffic features based on labelled data (Fig.1). In this research, we used the KDD CUP'99 [7] data set for this purpose, since all the existing approaches involved in the eomparison used the same data set for evaluation [13,14]. The full feature set assigned to this data set consists of 41 features. It is well known [15,16] that the KDD CUP'99 data set has several drawbacks regarding its suitability for representation of modern traffic. To avoid problems related to this data set, we split it into 4 parts according to the category of attack: DoS, Probe, U2R and R2L; we consider only two attack classes: DoS and Probe. This ensures more objective classification, since in such a way the influence of difference in eardinality of these subsets in the overall data set is reduced. We compare the feature-selection algorithms by the number of selected features as well as by the classification accuracy of machine learning algorithms chosen as classifiers for intrusion detection. Experimental results obtained over the KDD CUP'99 data set show that the $GeFS$ method outperforms the existing approaches by removing more than 30% of redundant features from the original data set, while keeping or yielding an even better classification accuracy. Even though the KDD CUP'99 data set does not refleet completely the characteristies of contemporary traffic, the results of our comparison indicate that the $GeFS$ method for selecting features would behave well on general intrusion detection data as well.

The paper is organized as follows. In Section 2, we give an overview of the feature-selection methods involved in the comparison. In Section 3, we present experimental setting as well as experimental results regarding the number of selected features and the classification accuracy obtained over the KDD Cup'99 data set. Section 4 summarizes our findings.

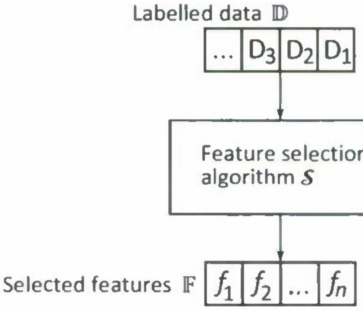

Labelled data 𝔻
... D3 D2 D1
Feature selection algorithm 𝑆
Selected features 𝔽 f1 f2 ... fn

Fig. 1. A feature selection algorithm

2 Feature-Selection Methods for Intrusion Detection

In this section, we first describe the previously known feature-selection methods used in intrusion detection. Then we give an overview of the recently proposed generic-feature-selection (*GeFS*) method together with 2 instances applied in intrusion detection: the correlation-feature-selection ($GeFS_{CFS}$) [11] and the minimal-redundancy-maximal-relevance ($GeFS_{mRMR}$) [12] measures.

2.1 Existing Approaches

2.1.1 SVM-Wrapper

Sung and Mukkamala [13] used the ranking methodology to select important features for intrusion detection: One input feature is deleted from the data at a time and the resultant data set is then used for the training and testing of the classifier Support Vector Machine (SVM) [17]. Then the SVMs performance is compared to that of the original SVM (based on all features) in terms of relevant performance criteria, such as overall accuracy of classification, training time and testing time. The deleted feature will be ranked as "important", "secondary" or "insignificant" according to the following rules:

- If accuracy decreases **and** training time increases **and** testing time decreases, **then** the feature is important.
- If accuracy decreases **and** training time increases **and** testing time increases, **then** the feature is important.
- If accuracy decreases **and** training time decreases **and** testing time increases, **then** the feature is important.
- If accuracy is not changed **and** training time increases **and** testing time increases, **then** the feature is important.
- If accuracy is not changed **and** training time decreases **and** testing time increases, **then** the feature is secondary.
- If accuracy is not changed **and** training time increases **and** testing time decreases, **then** the feature is secondary

- If accuracy is not changed **and** training time decreases **and** testing time decreases, **then** the feature is insignificant.
- If accuracy increases **and** training time increases **and** testing time decreases, **then** the feature is secondary.
- If accuracy increases **and** training time decreases **and** testing time increases, **then** the feature is secondary.
- If accuracy increases **and** training time decreases **and** testing time decreases, **then** the feature is insignificant

In [13] the experiment was conducted on a part of KDD CUP'99 data set [7]. This data set contains normal traffic and four main attack classes: Denial-of-Service (DoS) attacks, Probe attacks, User-to-Root (U2R) attacks and Remote-to-Local (R2L) attacks. Some important features were selected and the obtained data set after removing irrelevant features was classified by SVM [17]. The results are given in Table 1.

Table 1. Performance of SVM using selected features (SF) [13]

Classes	Number-of-SF	Accuracy
Normal	25	99.59%
DoS	19	99.22%
Probe	7	99.38%
U2R	8	99.87%
R2L	6	99.78%

2.1.2 Markov-Blanket

Markov blanket MB(T) of the output variable T is defined as the set of input variables such that all other variables are probabilistically independent of T. Knowledge of MB(T) is sufficient for perfectly estimating the distribution of T and thus for classifying T. Markov blanket has been applied for feature selection in many domains [4]. In 2004, Chebrolu et. al. [14] proposed to use Markov blanket for selecting important features for intrusion detection. In order to do that, they constructed a Bayesian Network (BN) from the original data set. A Bayesian network $B = (N, A, Q)$ is a Directed Acyclic Graph (DAG) (N, A) where each node $n \in N$ represents a domain variable (e.g. a data set attribute or variable), and each arc $a \in A$ between nodes represents a probabilistic dependency among the variables. A BN can be used to compute the conditional probability of one node, given values assigned to the other nodes. From the constructed BN, the Markov blanket of a feature T is the union of T's parents, T's children and eventually other parents of T's children. An example of a Bayesian Network is given in Fig.2. The gray-filled nodes constitute the MB(T):

For conducting the experiment, Chebrolu et. al. [14] randomly chose 11,982 instances from the overall (5 millions of instances) KDD CUP'99 data set [7]. 17 features were selected and the Bayesian Network [17] was used for classifying

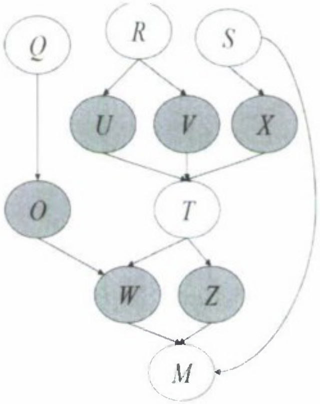


Fig. 2. An example of Markov blanket

Table 2. Performance of Bayesian Network using selected features (SF) [14]

Classes	Number-of-SF	Accuracy
Normal	17	99.64%
DoS	17	98.16%
Probe	17	98.57%
U2R	17	60.00%
R2L	17	98.93%

the obtained data set after removing irrelevant features. The results are given in Table 2.

2.1.3 CART

The Classification and Regression Trees (CART) approach [17] is based on binary recursive partitioning. The process is binary because parent nodes are always split into exactly two child nodes and recursive because it is repeated by treating each child node as a parent. The key elements of CART methodology are a set of splitting rules in a tree; deciding when the tree is complete and assigning a class to each terminal node. Feature selection for intrusion detection is based on the contribution of the input variables to the construction of the decision tree from the original data set. The importance of features is determined by the role of each input variable either as a main splitter or as a surrogate. Surrogate splitters are considered as back-up rules that closely mimic the action of primary splitting rules. For example, in the given model, the algorithm splits data according to the variable *protocol_type* and if a value for *protocol_type* is not available then the algorithm might use the *service* feature as a good surrogate. Feature importance, for a particular feature is the sum across all nodes in the tree of the improvement scores that the predictor has when it acts as a primary or surrogate splitter. For

Table 3. Performance of CART using selected features (SF) [14]

Classes	Number-of-SF	Accuracy
Normal	12	100%
DoS	12	85.34%
Probe	12	97.71%
U2R	12	64.00%
R2L	12	95.56%

example, for the node i, if the feature appears as the primary splitter then its importance could be given as $i_{importance}$. But if the feature appears as the n^{th} surrogate instead of the primary variable, then the importance becomes $i_{importance} = (p^n) \times i_{improvement}$ in which p is the *surrogate improvement weight* which is a user controlled parameter set between 0 and 1.

Chebrolu et. al. [14] conducted the experiment on the data set, which contains randomly chosen 11,982 instances from the overall (5 millions of instances) KDD CUP'99 data set [7]. 12 features were selected and the CART [17] was used for classifying the obtained data set after removing irrelevant features. The results are given in Table 3.

2.2 A New Generic-Feature-Selection Measure

In this subsection, we give an overview of the generic-feature-selection ($GeFS$) method together with 2 instances applied in intrusion detection: the ($GeFS_{CFS}$) and the ($GeFS_{mRMR}$) measures.

2.2.1 Definitions

Definition 1: A generic-feature-selection measure used in the so-called filter model is a function $GeFS(x)$, which has the following form [12]:

$$GeFS(x) = \frac{a_0 + \sum_{i=1}^{n} A_i(x)x_i}{b_0 + \sum_{i=1}^{n} B_i(x)x_i}, x = (x_1, \ldots, x_n) \in \{0,1\}^n \tag{1}$$

In this definition, binary values of the variable x_i indicate the appearance ($x_i = 1$) or the absence ($x_i = 0$) of the feature f_i; a_0, b_0 are constants; $A_i(x)$, $B_i(x)$ are linear functions of variables $x_1, \ldots, x_n$.

Definition 2: The feature selection problem is to find $x \in \{0,1\}^n$ that maximizes the function $GeFS(x)$ [12]:

$$\max_{x\in\{0,1\}^n} GeFS(x) = \frac{a_0 + \sum_{i=1}^{n} A_i(x)x_i}{b_0 + \sum_{i=1}^{n} B_i(x)x_i} \tag{2}$$

There are several feature selection measures, which can be represented by the form (1), such as the correlation-feature-selection (CFS) measure [1], the minimal-redundancy-maximal-relevance (mRMR) measure [2], Mahalanobis distance, etc.

A major challenge in the IDS feature-selection process is to choose appropriate measures that can precisely determine the relevance of features to the intrusion detection task and the redundancy between features. Since the relevance and the redundancy are usually characterized in terms of correlation or mutual information [4], the following measures for application in intrusion detection were considered in [11,12]: the correlation-feature-selection (CFS) measure [1] and the minimal-redundancy-maximal-relevance (mRMR) measure [2].

2.2.2 Correlation Feature Selection Measure

The Correlation Feature Selection (CFS) measure evaluates subsets of features on the basis of the following hypothesis: *"Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other"* [1]. The following equation gives the merit of a feature subset S consisting of k features:

$$Merit_S(k) = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

Here, $\overline{r_{cf}}$ is the average value of all feature-classification correlations, and $\overline{r_{ff}}$ is the average value of all feature-feature correlations. The CFS criterion is defined as follows:

$$\max_{S_k} \left[\frac{r_{cf_1} + r_{cf_2} + ... + r_{cf_k}}{\sqrt{k + 2(r_{f_1 f_2} + .. + r_{f_i f_j} + .. + r_{f_k f_1})}}\right] \quad (3)$$

Suppose that there are n full-set features. Binary values of the variable x_i are used to indicate the appearance ($x_i = 1$) or the absence ($x_i = 0$) of the feature f_i in the globally optimal feature set [11]. Therefore, the problem (3) can be rewritten as an optimization problem as follows:

$$\max_{x \in \{0,1\}^n} \left[\frac{(\sum_{i=1}^{n} a_i x_i)^2}{\sum_{i=1}^{n} x_i + \sum_{i \neq j} 2b_{ij} x_i x_j}\right] \quad (4)$$

It is obvious that the CFS measure is an instance of the GeFS measure. In [12], this measure was denoted by $GeFS_{CFS}$.

2.2.3 The mRMR Feature Selection Measure

In 2005, Peng et. al. [2] proposed a feature-selection method, which is based on mutual information. In this method, the relevance of features and the redundancy between features are considered simultaneously. In terms of mutual information, the relevance of a feature set S for the class c is defined by the mean value of all mutual information values between the individual feature f_i and the class c as follows:

$$D(S, c) = \frac{1}{|S|} \sum_{f_i \in S} I(f_i; c)$$

The redundancy between features in the set S is the mean value of all mutual information values between the feature f_i and the feature f_j:

$$R(S) = \frac{1}{|S|^2} \sum_{f_i, f_j \in S} I(f_i; f_j)$$

The mRMR criterion is a combination of two meansres given above and is defined as follows:

$$\max_S \left[\frac{1}{|S|} \sum_{f_i \in S} I(f_i; c) - \frac{1}{|S|^2} \sum_{f_i, f_j \in S} I(f_i; f_j)\right] \tag{5}$$

By using binary valnes of the variable x_i as in the case of the CFS measure to indicate the appearance or the absence of the featnre f_i and by denoting the mutual information values $I(f_i; c)$ and $I(f_i; f_j)$ by constants c_i and a_{ij}, respectively, the problem (5) can be described as an optimization problem as follows:

$$\max_{x \in \{0,1\}^n} \left[\frac{\sum_{i=1}^n c_i x_i}{\sum_{i=1}^n x_i} - \frac{\sum_{i,j=1}^n a_{ij} x_i x_j}{(\sum_{i=1}^n x_i)^2}\right] \tag{6}$$

It is also obvious that the mRMR measure is an instance of the GeFS measnre. In [12], this measnre was denoted by $GeFS_{mRMR}$.

Both the $GeFS_{CFS}$ and the $GeFS_{mRMR}$ feature-selection problems are solved by means of the techniqne that involves the Polynomial Mixed 0-1 Fractional Programming ($PM01FP$). The details are given below.

2.2.4 Polynomial Mixed 0-1 Fractional Programming

A general polynomial mixed $0-1$ fractional programming ($PM01FP$) problem [5] is represented as follows:

$$\min \sum_{i=1}^{m} \left(\frac{a_i + \sum_{j=1}^n a_{ij} \prod_{k \in J} x_k}{b_i + \sum_{j=1}^n b_{ij} \prod_{k \in J} x_k}\right) \tag{7}$$

$$\text{such that} \begin{cases} b_i + \sum_{j=1}^n b_{ij} \prod_{k \in J} x_k > 0, i = 1, .., m, \\ c_p + \sum_{j=1}^n c_{pj} \prod_{k \in J} x_k \leq 0, p = 1, .., m, \\ x_k \in \{0, 1\}, k \in J, \\ a_i, b_i, c_p, a_{ij}, b_{ij}, c_{pj} \in \Re. \end{cases}$$

By replacing the denominators in (7) by positive variables $y_i (i = 1, .., m)$, the $PM01FP$ then leads to the following equivalent polynomial mixed $0-1$ programming problem:

$$\min \sum_{i=1}^{m} \left(a_i y_i + \sum_{j=1}^{n} a_{ij} \prod_{k \in J} x_k y_i\right) \tag{8}$$

$$\text{such that} \begin{cases} b_i y_i + \sum_{j=1}^n b_{ij} \prod_{k \in J} x_k y_i = 1; y_i > 0, \\ c_p + \sum_{j=1}^n c_{pj} \prod_{k \in J} x_k \leq 0, p = 1, .., m, \\ x_k \in \{0, 1\}, k \in J, \\ a_i, b_i, c_p, a_{ij}, b_{ij}, c_{pj} \in \Re. \end{cases} \tag{9}$$

In order to solve this problem, Chang [5] proposed a linearization technique to transfer the terms $\prod_{k\in J} x_k y_i$ into a set of mixed $0-1$ linear inequalities. Based on this technique, the $PM01FP$ becomes then a mixed $0-1$ linear programming ($M01LP$), which can be solved by means of the branch-and-bound method to obtain the globally optimal solution.

Proposition 1: A polynomial mixed $0-1$ term $\prod_{k\in J} x_k y_i$ from (8) can be represented by the following program [5], where M is a large positive value:

$$\min z_i$$

$$\textit{such that} \begin{cases} z_i \geq 0, \\ z_i \geq M(\sum_{k\in J} x_k - |J|) + y_i \end{cases} \tag{10}$$

Proposition 2: A polynomial mixed $0-1$ term $\prod_{k\in J} x_k y_i$ from (9) can be represented by a continuous variable v_i, subject to the following linear inequalities [5], where M is a large positive value:

$$\begin{cases} v_i \geq M(\sum_{k\in J} x_k - |J|) + y_i, \\ v_i \leq M(|J| - \sum_{k\in J} x_k) + y_i, \\ 0 \leq v_i \leq M x_i, \end{cases} \tag{11}$$

The feature selection problem (2) is formulated as a polynomial mixed $0-1$ fractional programming ($PM01FP$) problem as follows:

Proposition 3: The feature selection problem (2) is a polynomial mixed $0-1$ fractional programming ($PM01FP$) problem.

Remark: By applying Chang's method [5], this $PM01FP$ problem can be transformed into an $M01LP$ problem. The number of variables and constraints is quadratic in the number n of full set features. This is because the number of terms $x_i x_j$ in (2), which are replaced by the new variables, is $n(n+1)/2$. The branch-and-bound algorithm can then be used to solve this $M01LP$ problem. But the efficiency of the method depends strongly on the number of variables and constraints. The larger the number of variables and constraints an $M01LP$ problem has, the more complicated the branch-and-bound algorithm is.

In [11,12], an improvement of the Chang's method was proposed in order to get an $M01LP$ problem in which the number of variables and constraints is linear in the number n of full set features. Details of the improvement are given below:

2.2.5 Optimization of the GeFS Measure

By introducing an additional positive variable, denoted by y, the following problem equivalent to (2) is considered:

$$\min_{x\in\{0,1\}^n} (-GeFS(x)) = -a_0 y - \sum_{i=1}^{n} A_i(x) x_i y \tag{12}$$

$$\textit{such that} \begin{cases} y > 0, \\ b_0 y + \sum_{i=1}^{n} B_i(x) x_i y = 1 \end{cases} \tag{13}$$

This problem is transformed into a mixed 0-1 linearning programming problem as follows:

Proposition 4: A term $A_i(x)x_iy$ from (12) can be represented by the following program, where M is a large positive value [12]:

$$\min z_i$$

$$\textit{such that} \begin{cases} z_i \geq 0, \\ z_i \geq M(x_i - 1) + A_i(x)y, \end{cases} \tag{14}$$

Proposition 5: A term $B_i(x)x_iy$ from (13) can be represented by a continuous variable v_i, subject to the following linear inequality constraints, where M is a large positive value [12]:

$$\begin{cases} v_i \geq M(x_i - 1) + B_i(x)y, \\ v_i \leq M(1 - x_i) + A_i(x)y, \\ 0 \leq v_i \leq Mx_i \end{cases} \tag{15}$$

Each term x_iy in (14), (15) is substituted by new variable t_i satisfying constraints from Proposition 2. Then the total number of variables for the $M01LP$ problem will be $4n + 1$, as they are x_i, y, t_i, z_i and $v_i(i = \overline{1, n})$. Therefore, the number of constraints on these variables will also be a linear function of n. As we mentioned above, with Chang's method [5] the number of variables and constraints depends on the square of n. Thus the method [11,12] actually improves Chang's method by reducing the complexity of the branch and bound algorithm.

3 Experimental Results

3.1 Experimental Setting

For comparison of the generic-feature-selection ($GeFS$) measure for intrusion detection [11,12] with the previously known ones [13,14], we implemented the $GeFS_{CFS}$ and the $GeFS_{mRMR}$ algorithms. The goal was to find globally optimal feature subsets by means of these two measures. Since different intrusion detection systems used different feature-selection methods and different classifiers with the aim of achieving the best classification results, we compared general performance of intrusion detection systems in terms of numbers of selected features and the classification accuracies of the machine learning algorithms giving the best classification results. For our experiment, we used the decision tree algorithm C4.5 [8] as classifier for the full-set data as well as for the data sets obtained by removing irrelevant features by means of the $GeFS_{CFS}$ and $GeFS_{mRMR}$ measures.

We performed our experiment using 10% of the overall (5 millions of instances) KDD Cup'99 data set [7], since all the existing approaches involved in the comparison used the same data set for evaluation [13,14]. This data set contains normal traffic (Normal) and four attack classes: Denial-of-Service (DoS),

Table 4. The partition of KDD CUP'99 data set used in the experiment

Classes	Number-of-instances	Percentage
Normal	97.278	18.35%
DoS	391.458	73.88%
Probe	41.113	7.77%
Total	529.849	100%

Probe, User-to-Root (U2R) and Remote-to-Local (R2L) attacks. As the two attack classes U2R and R2L have been criticized [15,16], we did not consider them for our experiment. Details of numbers of class instances are given in Table 4.

As the attack classes distribute so differently, the feature selection algorithm might concentrate only on the most frequent class data and neglect the others. Therefore, we chose to process these attack classes separately. In order to do that, we added normal traffic into each attack class to get two data sets: Normal&DoS and Normal&Probe. With each data set, we ran two feature-selection algorithms: the $GeFS_{CFS}$ and the $GeFS_{mRMR}$. The number of selected features is given in Fig.3. We then applied the C4.5 machine learning algorithm on each original full-set as well as each newly obtained data set that includes only those selected features from the feature-selection algorithms. We applied 5-fold cross-validation on each data set. The classification accuracies are given in Fig.4.

The $GeFS_{CFS}$ and the $GeFS_{mRMR}$ feature-selection methods were compared with the existing ones (the SVM-wrapper, the Markov-Blanket and the CART) regarding the number of selected features and regarding the classification accuracies of machine learning algorithms chosen as classifiers for intrusion detection process. Weka tool [3] that implements the machine learning algorithms (C4.5, SVM and BayesNet) was used for obtaining the results. In order to solve the $M01LP$ problem, we used TOMLAB tool [6]. All the obtained results are shown in Fig.3 and Fig.4.

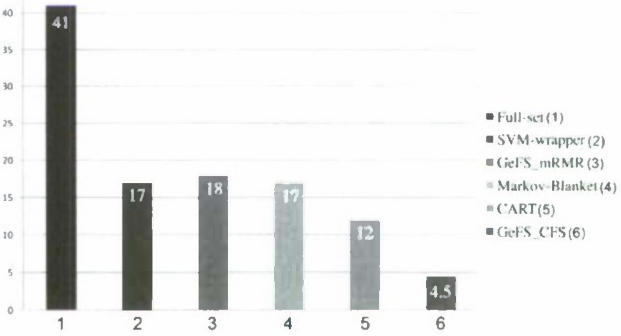


Fig. 3. Number of selected features (on average)

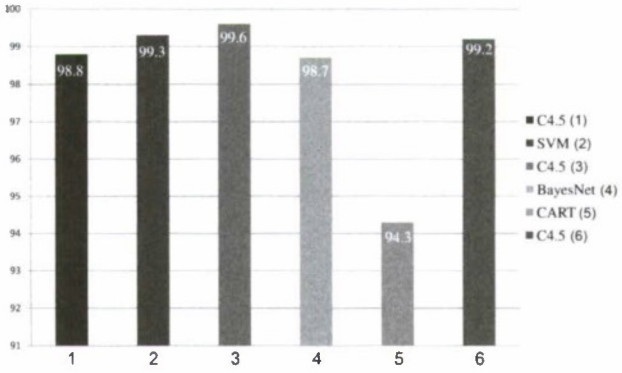


Fig. 4. Classification accuracies (on average)

3.2 Experimental Results

Fig.3 shows the average number of features selected by the *GeFS* feature-selection method and those selected by existing approaches. Fig.4 summarizes the average classification accuracies of chosen machine learning algorithms as classifiers for intrusion detection process. It can be observed from Fig.3 that the $GeFS_{CFS}$ feature-selection method selects the smallest number of relevant features. Fig.4 shows that with the approach from [11,12] the average classification accuracies are approximately the same or even better than those achieved by applying other methods.

4 Conclusions

In this paper, we compared, regarding the number of selected features and the classification accuracy, some previously known feature selection methods applicable for intrusion detection purposes with the feature selection methods for intrusion detection proposed in [11,12]. The previously known feature-selection algorithms involved in this comparison were the SVM-wrapper, Markov-blanket and CART algorithms. The feature selection algorithms proposed in [11,12] included in this comparison are instances of a generic-feature-selection (*GeFS*) method for intrusion detection: the correlation-feature-selection ($GeFS_{CFS}$) and the minimal-redundancy-maximal-relevance ($GeFS_{mRMR}$). Experimental results obtained over the KDD CUP'99 data set show that the *GeFS* method outperforms the previously known approaches by removing more than 30% of redundant features from the original data set, while keeping or yielding an even better classification accuracy. In spite of all the known limitations of the KDD CUP'99 data set used for comparison and the difficulties in establishing a more general theoretical basis for the comparison, there is a high probability that comparison results similar to ours could be obtained on other data sets as well.

References

1. Hall, M.: Correlation Based Feature Selection for Machine Learning. In: Doctoral dissertation. Department of Computer Science, University of Waikato (1999)
2. Peng, H., Long, F., Ding, C.: Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 1226–1238 (2005)
3. Weka, the Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/
4. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: Feature Extraction: Foundations and Applications. Studies in Fuzziness and Soft Computing. Springer, Heidelberg (2006)
5. Chang, C.T.: On the Polynomial Mixed 0-1 Fractional Programming Problems. European Journal of Operational Research 131, 224–227 (2001)
6. TOMLAB, The Optimization Environment in MATLAB, http://tomopt.com/
7. KDD Cup 1999 Data Set (1999), http://www.sigkdd.org/kddcup/index.php?section=1999&method=data
8. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
9. Gu, G., Fogla, P., Dagon, D., Lee, W., Skoric, B.: Towards an Information-Theoretic Framework for Analyzing Intrusion Detection Systems. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 527–546. Springer, Heidelberg (2006)
10. Crescenzo, G.D., Ghosh, A., Talpade, R.: Towards a Theory of Intrusion Detection. In: Capitani, S., Syverson, P., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 267–286. Springer, Heidelberg (2005)
11. Nguyen, H., Franke, K., Petrović, S.: Improving Effectiveness of Intrusion Detection by Correlation Feature Selection. In: International Conference on Availability, Reliability and Security (ARES), pp. 17–24. IEEE Press, New York (2010)
12. Nguyen, H., Franke, K., Petrović, S.: Optimizing a Class of Feature Selection Measures. In: NIPS 2009 Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra (DISCML), Vancouver, Canada (2009)
13. Sung, A.H., Mukkamala, S.: Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks. In: International Symposium on Applications and the Internet (SAINT), pp. 209–217. IEEE Press, Los Alamitos (2003)
14. Chebrolu, S., Abraham, A., Thomas, J.: Feature Deduction and Ensemble Design of Intrusion Detection Systems. Computers & Security 4, 295–307 (2005)
15. McHugh, J.: Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. ACM TISSEC 3, 262–294 (2000)
16. Sabhnani, M., Serpen, G.: Why Machine Learning Algorithms Fail in Misuse Detection on KDD Intrusion Detection Data Set. Intelligent Data Analysis 8, 403–415 (2004)
17. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley& Sons, New York (2001)
18. Chen, Y., Li, Y., Cheng, X.Q., Guo, L.: Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System. In: Lipmaa, H., Yung, M., Lin, D. (eds.) Inscrypt 2006. LNCS, vol. 4318, pp. 153–167. Springer, Heidelberg (2006)
19. Liu, H., Motoda, H.: Computational Methods of Feature Selection. Chapman & Hall/CRC, Boca Raton (2008)

From NLP (Natural Language Processing) to MLP (Machine Language Processing)

Peter Teufl[1], Udo Payer[2], and Guenter Lackner[3]

[1] Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology
peter.teufl@iaik.tugraz.at
[2] CAMPUS02, Graz University of Applied Science
udo.payer@campus02.at
[3] Studio78, Graz
guenther.lackner@studio78.at

Abstract. Natural Language Processing (NLP) in combination with Machine Learning techniques plays an important role in the field of automatic text analysis. Motivated by the successful use of NLP in solving text classification problems in the area of e-Participation and inspired by our prior work in the field of polymorphic shellcode detection we gave classical NLP-processes a trial in the special case of malicious code analysis. Any malicious program is based on some kind of machine language, ranging from manually crafted assembler code that exploits a buffer overflow to high level languages such as Javascript used in web-based attacks. We argue that well known NLP analysis processes can be modified and applied to the malware analysis domain. Similar to the NLP process we call this process Machine Language Processing (MLP). In this paper, we use our e-Participation analysis architecture, extract the various NLP techniques and adopt them for the malware analysis process. As proof-of-concept we apply the adopted framework to malicious code examples from Metasploit.

Keywords: Natural Language Processing, Malware Analysis, Semantic Networks, Machine Language Processing, Machine Learning, Knowledge Mining.

1 Introduction

Natural Language Processing (NLP) involves a wide range of techniques that enable the automated parsing and processing of natural language. In the case of written text, this automated processing ranges from the lexical parsing of sentences to applying sophisticated methods from machine learning and artificial intelligence in order to gain insight on the covered topics. Although NLP is a complex and computationally intensive task, it gains more and more importance due to the need to automatically analyze large amounts of information stored within arbitrary text sources on the Internet. For such large text corpora it is

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 256–269, 2010.

not feasible for human experts to read, to understand and to draw conclusions in a complete manual way.

An example for such a domain is the electronic participation (further denoted as e-Participation) of citizens within a governmental decision process. Typically, this process involves citizens that express their opinion on certain topics and domain experts that analyze these opinions and extract important concepts and ideas. In order to speed up the process and improve the results it makes sense to apply NLP techniques that support the domain experts. Therefore, we have implemented and employed an e-Participation analysis framework [1].

Due to previous work in the field of malicious code detection—*especially in the field of polymorphic shellcode detection* [2], [3] — we realized that the analysis of natural languages is somewhat similar to the analysis of machine languages. Malware, regardless of its nature, is always based on some kind of programming language used to encode the commands that an attacker wants to execute on a victim's machine. This can be raw assembler code or a high level scripting language such as Javascript. The process of detecting malware is to identify malicious code within large amounts of regular code. There are a wide range of malware detection methods ranging from simple signature detection methods to highly sophisticated methods based on machine learning. However, before such methods can be deployed for malware detection we need to analyze and understand the underlying code itself. Due to self mutating code, encryption, metamorphic and polymorphic engines, and other methods designed to camouflage the malware itself, it is not possible to create simple signatures anymore. Therefore, we need to extract other more complex relations within the machine language that allow us to devise more robust detection methods.

In this paper, we argue that the same NLP processes and techniques used for the analysis of natural language can be mapped and applied to machine language. Analog to the NLP process we introduce the concept of Machine Language Processing (MLP). In order to find relevant MLP processes, we extract the various analysis steps of our e-Participation analysis framework and define corresponding MLP steps. In order to test the implementability of this approach we finally apply the modified framework to real assembler code extracted from various decoding engines generated by the Metasploit framework.

Although the proof-of-concept and the NLP-to-MLP transformations focus on assembler code, the discussed techniques could easily be extended to arbitrary machine languages.

2 Related Work

Malware is defined as some piece of software with the only intention to perform some harmful actions on a device, which is already under control or is intended to be under control of an attacker. Malware analysis—*on the contrary*—is the process of re-engineering these pieces of software or to analyze the behavior for the only purpose to identify or demonstrate the harmfulness of these pieces of software (such as a virus, worm, or Trojan horses). Actually, malware analysis can be divided into

- behavior analysis (*dynamic analysis*) and
- code analysis (*static analysis*)

Since no generic tool exists to perform this analysis automatically, the process of malware analysis is a manual one, which can fortunately fall back on a rich set of efficient but simple tools. A tricky part in malware analysis is to detect pieces of code, which are only triggered under some specific conditions (day, time, etc. ...). In such cases, it is essential to disassemble the whole executable and to analyze all possible execution pathes. Finding and watching such execution pathes (e.g. by the help of a disassembler) is forming the core mechanism of a sophisticated code analysis process.

As "dynamic" approach to detect execution chains within a piece of software is to execute and analyze its behavior in a restricted environment. Such an environment can be a debugger, which is controlled by a human analyst, to step through each single line of code to see the code-execution happening and to understand the "meaning" of the code. Examples of such "sandbox"- techniques are CWSandbox [4], the Norman SandBox [5], TTAnalyze and Cobra [6]. Common to all these examples is that code is automatically loaded and analyzed in a virtual machine environment to find out the basic behavior and execution pathes. A special dynamic sandbox-method is the so called black box analysis. In this case, the system is studied without any knowledge about its internal construction. Observable during the analysis are only external in- and outputs as well as their timing relationships. After a successful simulation, a post mortem analysis will show effects of the malware execution. This post mortem analysis can be done by standard computer forensic tool chains.

In the case of malicious code analysis, the common idea is to use *analysis archtitectures* to make use of the huge number of useful tools in a controlled way. BitBlaze [7] for instance even tries to combine static- and dynamic analysis tools. The BitBlaze framework actually consists of three components: Vine, the static analysis tool, TEMU, the dynamic analysis component, and Rudder, a separate tool to combine dynamic and static analysis results.

NLP is a huge field in computer science about language- based interactions between computers and humans. It can basically be divided in the following two major areas:

- *Natural language generation systems* (LGS), which convert information from computer databases into readable human language and
- *Natural language understanding systems* (LUS), which are designed to convert samples of human language into a formal representation. Such a representation can be used to find out what concepts a word or phrase stands for and how these concepts fit together in a meaningful way.

Related to the content of this paper, we always think about NLP as an application that can deal with text in the sense of classification, automatic translations, knowledge acquisition or the extraction of useful information. In this paper, we will not link NLP to the generation of natural languages. Especially in the case of LUS, a lot of prior work exists, which was carried out by many different research groups (e.g. [8],[9]). Machine learning techniques have been applied to the natural language problem, statistical analysis has been performed and large text corpora have been generated and have been used successfully in the field of NLP. Thus, several projects—*about innovative ways to run and improve NLP-methods*—have already been finished or are still ongoing - and we are quite sure that there will be many more.

3 Methods

3.1 NLP

All NLP components of the platform are based on the lingpipe NLP API [10]. It is a Java API that covers a wide range of algorithms and techniques important for NLP: Examples are Part-of-Speech (POS) tagging, the detection of sentences, spelling correction, handling of text corpora, language identification, word sense disambignation (e.g. [11]), etc. The techniques that are relevant for our text-analysis architecture will be shortly discussed in the subsequent sections. For a good overview of all these techniques we refer to the tutorials that come with the lingpipe package[1].

3.2 Semantic/Associative Networks and Spreading Activation (SA)

Associative networks [12] are directed or undirected graphs that store information in the network nodes and use edges (links) to present the relation between these nodes. Typically, these links are weighted according to a weighting scheme. Spreading activation (SA) algorithms [13] can be used to extract information from associative networks. Associative networks and SA algorithms play an important role within Information Retrieval (IR) systems such as [14], [15] and [11]. By applying SA algorithms we are able to extract *Activation Patterns* from trained associative networks. These *Activation Patterns* can then be analyzed by arbitrary supervised and unsupervised machine learning algorithms.

3.3 Machine Learning (ML)

For the supervised or unsupervised analysis of the *activation patterns* – the patterns generated by applying SA to the semantic/associative network – standard machine learning algorithms can be applied. Examples for supervised algorithms are the widely used Support Vector Machines (SVM), Neural Networks and Bayesian Networks. The family of unsupervised algorithms has an important role, since such techniques allow us to extract relations between features, to detect anomalies and to find similarities between patterns without having an a-priori knowledge about the analyzed data. Examples for such algorithms are Neural Gas based algorithms [16], Self Organizing Maps (SOM), Hierachical Agglomerative Clustering (HAC), or Expectation Maximation (EM). In this work we employ the Robust Growing Neural Gas algorithm (RGNG) [16].

[1] http://alias-i.com/lingpipe/demos/tutorial/read-me.html

4 From NLP to MLP

In [1] we present an automated text-analysis architecture that is used for the analysis of various e-Participation related data-sets. The basic modules of this architecture are depicted in Figure 1. The remaining part of this section describes the various NLP and ML related submodules of this architecture and how they can be applied or transformed to MLP modules for malware analysis.

4.1 Lexical Parser/Emulator/Disassembler

NLP: For NLP, we need to convert a sequence of characters into a sequence of tokens. These tokens represent the terms of the underlying text. The conversion process is called lexical analysis. By using lexical parsers such as the Stanford Parser [17], we are able to extract the roles of terms within a sentence and the relations between these terms. Depending on the subsequent processing steps, this could range from a superficial analysis identifying some key grammatical concepts to a deep analysis that is able to extract fine details.

MLP: Raw machine code is a byte sequence that contains instructions that are executed by the processor. In addition, most of the available instructions have parameters that are also encoded in the byte sequence. In order to extract information for further analysis, we need to process this byte sequence and extract the instructions and the parameters. In a simple scenario this could be done

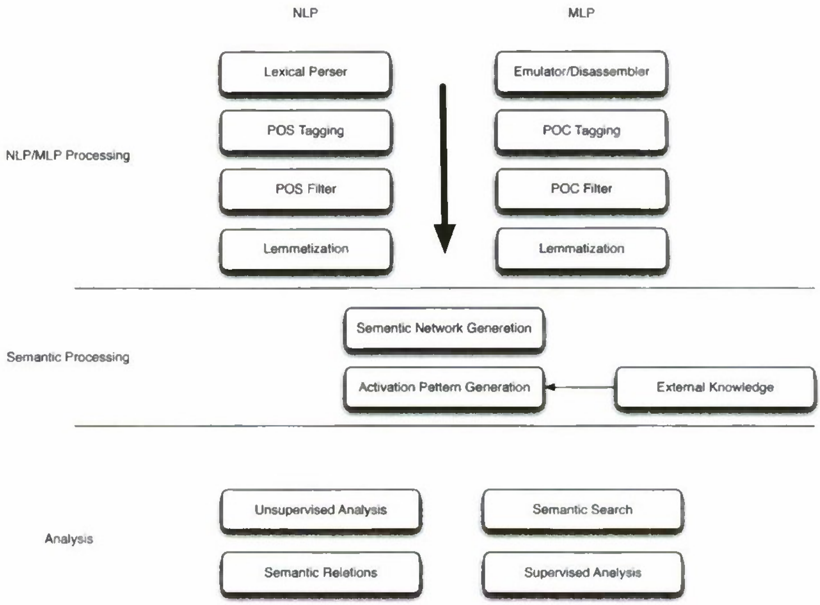


Fig. 1. MLP vs. NLP Processing

with a disassembler that extracts instructions from a given byte sequence. However, due to branch operations such as **jmp** or **call** these byte sequence is not processed by the CPU in a linear way. Thus, in order to extract the instruction chain the way it is executed on a CPU, we need to employ emulators or execute the code directly on the CPU. For the example presented later in this work, we utilize the PTRACE system call[2] on linux to execute code directly on the CPU (see Section 5.1 for a more detailed description). By applying such methods to the raw byte sequence, we are able to extract and inspect the instructions chains executed on the CPU. In analogy to NLP these instruction chains represent the written text, which needs to be analyzed. Similar to NLP the deepness of the analysis depends on the applied method. These methods range from extracting the instructions and their execution order to more complicated methods capable of identifying more complex structures: constructs such as loops, the necessary preparation for executing interrupts, branching etc.

4.2 POS (Part-of-Speech) Tagging, POC (Part-of-Code) Tagging

NLP: POS tagging can also be seen as part of the lexical analysis described in the previous section. However, since it plays an important role for text analysis, we describe it as separate process. In NLP, Part-of-Speech tagging is the process of identifying the role of each term in a sentence. The following example shows the POS tags for a given sentence: *Hello_RB I_PRP am_VBP a_DT little_JJ sentence_NN trying_VBG to_TO find_VB my_PRP place_NN within_IN this_DT text_NN* ._. The tags were obtained by using the online interface of the Stanford parser[3], where for example *NN* indicates nouns and *VB** identifies verbs and their different modes. POS tags are used for subsequent processing steps, which include the filtering of terms according to their tags and establishing relations between terms in a semantic network according to these tags.

MLP: Obviously, there are no nouns, verbs or related concepts in machine code, but there are similar concepts that could be used to tag single instructions. We call these tags Part-Of-Code (POC) tags. For the example presented in Section 5 we tag the instructions according to their functionality which results in the following categories: control flow, arithmetic, logic, stack, comparison, move, string, bit manipulation, flag manipulation, floating point unit instructions, other.

4.3 POS/POC Filtering

NLP: Depending on the subsequent analysis, it makes sense to keep only terms with certain POS tags. For the e-Participation related text analysis, we only keep nouns, verbs and adjectives since the already convey a large part of the information within the text.

[2] http://linux.die.net/man/2/ptrace

[3] http://nlp.stanford.edu:8080/parser/

MLP: According to the determined POC tags, we can easily define filters that allow us to focus on branching behavior, arithmetic operations, logical operations etc.

4.4 Lemmatization

NLP: Before proceeding with the NLP analysis of POS tagged text, it makes sense to derive the lemmas of the remaining terms. By doing so we avoid the ambiguity of different forms such as inflected terms or plural forms. For example the term **bought** would be mapped to its lemma **buy** for further analysis.

MLP: When applying this process to machine code, we need to ask "What is the lemma of an assembler instruction?". There is not a single answer to this question, but there are several concepts that could be used for lemmatization:

- **Instruction without parameters:** In this case we strip away the parameters of an instruction and use the instruction as lemma.
- **Mapping of instructions:** Instructions that belong to the same family could be mapped to one instruction. An example would be the mapping of all **mov** derivates to one instruction.
- **High level interpretation:** In this case we focus on the operations performed by the instructions and not the instructions themselves. E.g. the instructions and their parameters **xor eax,eax** or **mov eax,0** or the chain **mov eax,5; sub eax,5** all have the same effect – the **eax** register contains the value 0. As we see, this effect can be achieved by using various instructions or instruction chains. Such techniques are typically employed by polymorphic and metamorphic engines trying to camouflage their real purpose by changing the signature of each generated instance.

4.5 Creation of the Associative/Semantic Network

In this step we create the semantic or associative network that stores the information on how different features are related. In case of NLP, the terms of a text are the features and the relations are defined by the co-occurence of terms within sentences. For MLP, the features are represented by instructions and the relations between instructions are based on the co-occurence of these instructions within chains. We note that although these relations are rather simple they already convey important information for further analysis (see Section 6 for possible improvements). The semantic network is generated in the following way:

NLP: For each sentence, we apply the following procedure: For each different term (sense) within the analyzed text corpus we create a node within the associative network. The edges between nodes and their weights are determined in the following way: All senses within a sentence are linked within the associative network. Newly generated edges get an initial weight of 1. Every time senses co-occur together, we increase the weight of their edges by 1. In addition, we store the type of connection for each edge. Examples for these types are noun-to-noun links, noun-to-verb links or adjective-to-adverb links. By using this information when applying SA algorithms, we are able to constrain the spreading of activation values to certain types of relations.

MLP: In machine code, sentences as we know them from text, do not exist. However we can find other techniques that separate instruction chains in a meaningful way:

- **Using branch operations to limit instruction chains:** For this method, we use branch operations such as **jmp**, **call** to identify the start/end of an instruction chain. We have already sucessfully applied this method in prior work ([3]).
- **Number of instructions:** We could simply define a window with size n that take n instructions from the instruction chains.

Regardless of the method for the extraction of instruction chains, the network is generated in the same way as for the text data.

4.6 Generation of Activation Patterns

Information about the relations between terms/instructions can be extracted by applying the SA-algorithm to the network. For each sentence/instruction chain, we can determine the corresponding nodes in the network representing the values stored in the data vector. By activating these nodes and applying SA, we can spread the activation according to the links and their associated weights for a predefined number of iterations. After this process, we can determine the activation value for each node in the network and represent this information in a vector - the *Activation Pattern*. The areas of the associative network that are activated and the strength of the activation gives information about which terms/instructions occurred and which nodes are strongly related.

4.7 Analysis of Activation Patterns

The *activation patterns* generated in the previous layers are the basis for applying supervised and unsupervised Machine Learning algorithms. Furthermore, we can implement semantic aware search algorithms based on SA.

Unsupervised Analysis: Unsupervised analysis plays an important role for the analysis of text, since it allows us to automatically cluster documents or instruction chains according to their similarity.

Search with Spreading Activation (SA): In order to search for related concepts within the analyzed text sources/instruction chains, we apply the following procedures:

1. The user enters the search query, which could be a combination of terms or instructions, a complete sentence or instruction chain or even a document containing multiple sentences or instruction chains.
2. We determine the POS/POC tags for every term/instruction within the search query.
3. Optionally, we now make use of an external knowledge source to find related terms/instructions and concepts for the terms/instructions in the query. For NLP such an external source could be WordNet [14] or Wikipedia. For MLP we could use reference documentation that describes all available instructions, their parameters and how these are related. An example for such a source is the Instruction Set Reference for Intel CPUs[4].
4. We activate the nodes corresponding to the terms/instructions of the search query and use the SA algorithm to spread the activation over the associative network.
5. We extract the *activation pattern* of the associative network and compare it to the document, sentence or instruction chain patterns that were extracted during the training process. The patterns are sorted according to their similarity with the search pattern.

External knowledges sources such as Wordnet can be quite useful for improving the quality of the search results. In order to highlight some of the benefits, we have the following example for text-analysis. Assuming, we execute a search query that contains the term **fruit**. After applying SA, we get the relations that were generated during the analysis of the text. However, these relations only represent the information stored within the text. The text itself does not explain that apples, bananas and oranges are instances of the term **fruit**. Therefore when searching for **fruit** we will not find a sentence that contains the term **apple** if the relation between these two terms is not established within the text. Thus, it makes sense to include external knowledge sources that contain such information. For NLP we can simply use Wordnet to find the instances of **fruit** and activate these instances in the associative network before applying SA. For MLP, such information could also provide vital information about the relations between instructions. In a similar way we could issue a search query that extends the search to all branch or arithmetic instructions.

Relations between Terms/Instructions: The trained associative network contains information about relations between terms/instructions that co-occur within sentences/instruction chains. By activating one or more nodes within this network and applying the SA algorithm, we are able to retrieve related terms/instructions.

5 The Real World – Example

In order to show the benefits of a possible malware analysis architecture based on MLP, we transform the existing NLP framework and apply it to payloads

[4] http://www.intel.com/products/processor/manuals/

and shellcode encoders generated by the Metasploit framework. The Metasploit project is described in this way on the project website[5]: *Metasploit provides useful information to people who perform penetration testing, IDS signature development, and exploit research. This project was created to provide information on exploit techniques and to create a useful resource for exploit developers and security professionals. The tools and information on this site are provided for legal security research and testing purposes only.*

5.1 PTRACE Utility

For the lexical analysis of an arbitrary byte sequence we have developed a simple tool based on the PTRACE system call[6] on Linux.

- **Single stepping:** By utilizing PTRACE we are able to instruct the processor to perform single stepping. This enables us to inspect each executed instruction, its parameters and the CPU registers.
- **Execution of arbitrary byte sequences:** The utility follows each instruction chain until the bounds of the byte sequence are reached, the maximum number of loops is reached or a fault occurs. Whenever one of these conditions is fulfilled, the tool searches for a new entry point that has not already been executed. By applying these technique we are able to find executable instruction chains even if they are embedded in other data (e.g. images, network traffic).
- **Blocking of interrupts:** The analysis of the payloads and encoders generated by Metasploit is rather simple. In order to keep payloads from writing on the harddrive, we simple block all interrupts encountered by the tool.
- **Detection of self modifying code:** Such behavior is typical for a wide range of encoders/decoders that encode the actual payload in order to hide it from IDS systems. Typically the actual payload is decoded (or decrypted) by a small decoder. After this process the plain payload is executed. Since this decoding process changes the byte sequence, it is easy to detect when the decoder has finished and jumps into the decoded payload.
- **Dumping of instructions:** The tool makes use of the libdisasm library[7] to disassemble instructions. For each CPU step, we dump the instruction, its parameters and the category it belongs to.

5.2 Metasploit Data

Metasploit offers a command line interface to generate and encode payloads. We have used this interface to extract various payloads. Furthermore, we have encoded a payload with different shellcode encoders including the polymorphic shellcode encoder shikata-ga-nai. As dump format we have used the unsigned char buffer format. In order to apply MLP techniques we use the existing NLP architecture as basis and add or modify existing plugins for MLP processing:

[5] http://www.metasploit.com/

[6] http://linux.die.net/man/2/ptrace

[7] http://bastard.sourceforge.net/libdisasm.html

- **Lexical Analysis:** For the extraction of the instruction chain we use our ptrace utility. The extracted chains contain the executed instructions, their parameters and the instruction category. We do not consider the parameters for further processing. The instruction chains are seperated into smaller chains by using control flow instructions (e.g. **jmp, call, loop**) as separator. In analogy to NLP, these sub instruction chains are considered as "sentences" whereas the whole payload/encoded payload is considered as "document".
- **Tagging:** Similar to a POS tagger, we can use a POC tagger for MLP. In this case this tagger uses the instruction category as tag. We consider all tags for further analysis and do not apply a filter.
- **Lemmatization:** Except for dropping the parameters, we do not employ further lemmatization operations.
- **Semantic network generation:** We apply the same semantic network generation process as used in the NLP architecture.
- **Activation pattern generation:** This is also based on the same process that is used for the NLP architecture. For each sub instruction chain (sentence), we activate the nodes corresponding to the instructions within the chain and spread the activation over the semantic network. We do not make use of any external knowledge source.
- **Analysis:** We show some examples for the analysis of the extracted/encoded payloads: Unsupervised clustering, finding relations between instructions and semantic search.

5.3 Relations

For text-analysis we often need to find terms that are closely related to a given term. An example from the e-Participation data analysis is shown in Figure 2(a). We use the term **vehicle** and extract the related terms from the the semantic network. Some examples for related terms are: **pollution**, **climate change**, **car**, **pedestrian** and **pedestrian crossing**. These relations are stored in the semantic network that was generated during the analysis of the text data. In MLP, we can apply exactly the same procedure. For the following example we want to find instructions that are related to **XOR** within the dataset consisting of subchains. In this case relation means that the instructions co-occur within the same chain. By issuing the query for **xor**, we get the following related instructions: **push, pop, inc, add, dec, loop**. These results can be explained by having a closer look on the decoding loops of various decoders (shikata-ga-nai, countdown, alpha-mixed) shown in Table 1. The utilitzation of these other instructions is necessary for reading the encoded/encoded shellcode, performing the actual decoding and writing the decoded shellcode back onto the stack. Due to the unsupervised analysis and the semantic network we are able to find these relations without knowing details about the underlying concepts.

5.4 Semantic Search

The previous example shows that due to the semantic network and the links within this network we are able to find relations between terms/instructions.

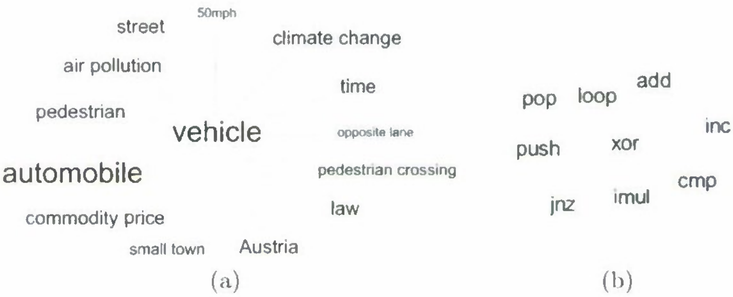


Fig. 2. NLP - Relation between terms (a) and MLP - relations between instructions (b)

Table 1. Semantic search results for instruction **add**

Result	Decoder	Instruction chain	Description
1	shikata-ga-nai	xor add add loop	Decoder
2	shikata-ga-nai	xor mov fnstenv pop mov xor add add loop	Decoder setup
3	nonalpha	pop mov add mov cmp jge	Decoder setup
4	fnstenv-mov	xor sub loop	Decoder
5	countdown	xor loop	Decoder

These relations can also be used for executing semantic aware search queries. In order to highlight the benefits, we first present a simple example taken from text-analysis. Assuming we have two sentences[8] **A** and **B**: **A**: "Evidence suggests flowing **water** formed the rivers and gullies on the **Mars** surface, even though surface temperatures were below freezing" and **B:** "Dissolved minerals in liquid **water** may be the reason". When we search for the term **Mars** we obviously are able to retrieve sentence **A**. However, since sentence **A** talks about water on Mars, we also want to find sentence **B** that adds further details concerning the term **water**. Since the term **Mars** is not in sentence **B** we need to make use of the relations stored in the semantic network in order to include sentence **B** in the search results. The same procedure can be applied to MLP. For the following example we search for instruction chains that are related to the instruction **add**, which plays a role in various shellcode decoders. The results are shown in Table 1. Obviously, the algorithm returns decoders with an **add** instruction first, since these have the best matching pattern. However, at position 4 and 5 we also retrieve decoding loops of other decoders that do not make use of the **add** instruction. We are able to find these decoding loops since they use other instructions that are typical for such loops: **xor, sub, loop**. Due to the relations created by the decoding loops of shikata-ga-nai, **add** is linked with those and similar instructions. Thus, we are able to retrieve these other decoder loops that do not contain the **add** instruction, but have similar tasks.

[8] Take from the article: NASA Scientists Find Evidence for Liquid Water on a Frozen Early Mars, May 28th, http://spacefellowship.com

5.5 Clustering

By clustering whole execution chains or sub chains (e.g. loops) into clusters, we are able to categorize different execution chains automatically. For unsupervised clustering we apply the RGNG [16] cluster algorithm to the *activation patterns* of the subchain dataset. By choosing a rather simple model complexity, we retrieve 4 clusters: Cluster 1 primarily consists of the decoding loops of **alpha-upper** and **alpha-mixed**. Since both decoders have similar tasks (but not the same instruction chains), they are categorized within the same cluster. Cluster 2 and Cluster 4 contain the polymorphic decoding engines of **shikata-ga-nai**. By observing the instruction chains of those both clusters we see that Cluster 2 has chains based on **add** instructions whereas Cluster 4 consists of those chains that employ **sub** instructions. This is a perfect example why it could make sense to employ external knowledge to gain additional information about the analyzed instructions. In this case, **add** and **sub** could be mapped to arithmetic instructions which would result in the categorization within the same cluster. Cluster 3 contains chains related to decoding engine setup and the necessary preparations for calling an interrupt (typically the payload itself).

6 Conclusions and Outlook

In this paper we present a MLP architecture for malware analysis. This architecture is the result of adopting an existing NLP architecture to the analysis of machine code. We map existing NLP modules to MLP modules and describe how established NLP processes can be transferred to malware analysis. In order to show some of the possible applications for such an MLP architecture, we analyze different shellcode engines and payloads from the Metasploit framework. The presented malware architecture can be seen as the first step in this direction. There are further promising techniques, which would increase the capabilities and the quality of the analysis process:

- Improved lexical parsing in order to allow the identification of more complex structures such as loops, preparations for interrupts, etc.
- Due to improved lexical parsing, more relations could be stored in the semantic network, which would enable more detailed or focused analysis processes.
- High level interpretation of the underlying machine code.
- Extending the MLP framework to high level languages such as Javascript.

All of these suggested improvements have corresponding elements within NLP and are partly already solved there. This means, that we might be able to apply some of these techniques directly in MLP or adapt them for MLP. As next step we will identify more suitable NLP techniques and adopt them to MLP modules. Finally, we especially want to thank P. N. Suganthan for providing the Matlab sources of RGNG [16].

References

1. Teufl, P., Payer, U., Parycek, P.: Automated analysis of e-participation data by utilizing associative networks, spreading activation and unsupervised learning. In: Macintosh, A., Tambouris, E. (eds.) Electronic Participation. LNCS, vol. 5694, pp. 139–150. Springer, Heidelberg (2009)
2. Payer, U., Teufl, P., Kraxberger, S., Lamberger, M.: Massive data mining for polymorphic code detection. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 448–453. Springer, Heidelberg (2005)
3. Payer, U., Teufl, P., Lamberger, M.: Hybrid engine for polymorphic code detection. In: Julisch, K., Krügel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 19–31. Springer, Heidelberg (2005)
4. SunbeltSoftware (Cwsandbox - automatic behavior analysis of malware)
5. Norman, Norman sandbox: A virtual environment where programs may perform in safe surroundings
6. Vasudevan, A., Yerraballi, R.: Cobra: Fine-grained malware analysis using stealth localized-executions. In: IEEE Symposium on Security and Privacy, pp. 264–279 (2006)
7. Song, D., Brumley, D., Yin, H., Caballero, J., Jager, I., Kang, M.G., Liang, Z., Newsome, J., Poosankam, P., Saxena, P.: Bitblaze: A new approach to computer security via binary analysis. In: Sekar, R., Pujari, A.K. (eds.) ICISS 2008. LNCS, vol. 5352, pp. 1–25. Springer, Heidelberg (2008)
8. Microsoft, Natural language processing group: Redmond-based natural language processing group
9. Stanford, Natural language processing group: Natural language processing group at stanford university
10. Alias-i, Lingpipe: A suite of java libraries for the linguistic analysis of human language
11. Tsatsaronis, G., Vazirgiannis, M., Androutsopoulos, I.: Word sense disambiguation with spreading activation networks generated from thesauri. In: Veloso, M.M. (ed.) IJCAI 2007 (2007)
12. Quillian, M.R.: Semantic memory. MIT Press, Cambridge (1968)
13. Crestani, F.: Application of spreading activation techniques in information retrieval. Artificial Intelligence Review 11, 453–482 (1997)
14. Fellbaum, C.: WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press, Cambridge (1998)
15. Kozima, H.: Similarity between words computed by spreading activation on an english dictionary. In: EACL, pp. 232–239 (1993)
16. Qin, A.K., Suganthan, P.N.: Robust growing neural gas algorithm with application in cluster analysis. Neural Netw. 17, 1135–1148 (2004)
17. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Advances in Neural Information Processing Systems (NIPS), vol. 15, pp. 3–10. MIT Press, Cambridge (2002)

Secure Multi-Agent System for Multi-Hop Environments

Stefan Kraxberger, Peter Danner, and Daniel Hein

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria
{skraxberger,pdanner,dhein}@iaik.tugraz.at

Abstract. Multi-agent systems allow a multitude of heterogenous systems to collaborate in a simple manner. It is easy to provide and gather information, distribute work and coordinate tasks without bothering with the differences of the underlying systems. Unfortunately, multiple networking and security problems arise from the dynamic behavior of multi-agent systems and the distributed heterogeneous environments in which they are used. With our work we provide a solution enabling secure collaboration and agent execution as well as agent mobility in multi-hop environments. We achieve this by using a secure unstructured P2P framework as communication layer and integrate it with a well known multi-agent system.

Keywords: Security, Multi-Agent Systems, Multi-Hop Networks, Peer-to-Peer.

1 Introduction

Multi-agent systems (MAS) have been used to solve problems that are out of reach for stand-alone or monolithic systems. Examples of problems to which multi-agent systems have been applied include control systems [1], [2], timetable coordination [3], disaster response [4], [5]. MAS are especially promising for disaster response scenarios. Since the specific tasks of such scenarios like information gathering, on-demand computation, information distribution, and team coordination are well-suited for MAS.

In close relation to MAS we find the peer-to-peer (P2P) concept. Structured P2P systems are very prominent since they are well fitted for data storage and distribution. Their internal organization and function is optimized for addressing data in a distributed environment. Conversely, they are ill suited for the purpose of a general overlay network that provides general communication and resource sharing functions. Unstructured P2P networks are ideal for establishing a general overlay since they only provide the means of organizing the overlay topology and providing connectivity between the separate nodes.

Besides all the functions which have been enabled by multi-agent and peer-to-peer systems, new kinds of network security threats have been introduced [6, 7, 8, 9] as well. These new threats are much more difficult to address because

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 270–283, 2010.

of the composition and distributed nature of these systems. Malicious or selfish nodes can distract, disturb, obstruct and impede the correct execution of P2P systems often with little effort [10,11,12]. In a P2P system every entity is equal. Every entity provides the same set of functions. In a centralized system this is restricted to a select few. This fact necessitates global protection of all entities and their interactions.

The main contribution of our work is to first enable multi-agent systems to work in multi-hop environments and second to provide means to do that in a secure manner. Our work provides a comprehensive solution for building a Java-based multi-agent-system with a secure peer-to-peer communication layer. We used the two existing systems JADE and the Secure P2P framework (SePP), and integrated them using the JADE communication interface. Out of the box the communication in JADE is based on Remote Method Invocation (RMI), which only guarantees end-to-end security via SSL encryption. Our approach relies on a peer-to-peer system to guarantee not only authentication, integrity and confidentiality in direct-connected networks but also in multi-hop environments. In addition, SePP is based on a scalable security concept that allows to adjust the security measures according to the needs and capabilities of participating devices.

Subsequently, we present how the JADE agent middleware and the SePP framework can be combined to develop a secure multi-agent-system for multi-hop environments. We briefly outline the design and implementation of SePP. Thereafter, we present the messaging in Jade and the default network implementation. Our implementation is concisely described with a focus on the specific solutions such as the message dispatching or the transparent proxy generation. At last we provide results and a short benchmark which compares vanilla JADE and our implementation.

2 Motivation

MAS technology relies heavily on the existence of a network infrastructure. Unfortunately, in case of an emergency it can not be assumed that an infrastructure bound network is fully working. Emergencies can occur in isolated regions which lack the necessary infrastructure such as comprehensive wired or wireless network coverage. Also, a disaster which caused the emergency could have destroyed or disrupted the required infrastructure. The absence of a functioning static network infrastructure necessitates that crucial information for on-site emergency response must be made available through mobile ad-hoc networks. Ideally, this mechanism is backed by fall-back communication facilities such as GSM, UMTS, satellite communication, and TETRA. The information required to respond properly in case of emergencies usually consists of confidential data including personal health records or electric grid maps that should only be available to authorized personnel. Thus, the provision of network connectivity, as well as managing access to confidential data during the emergency response operation is a substantial network security challenge.

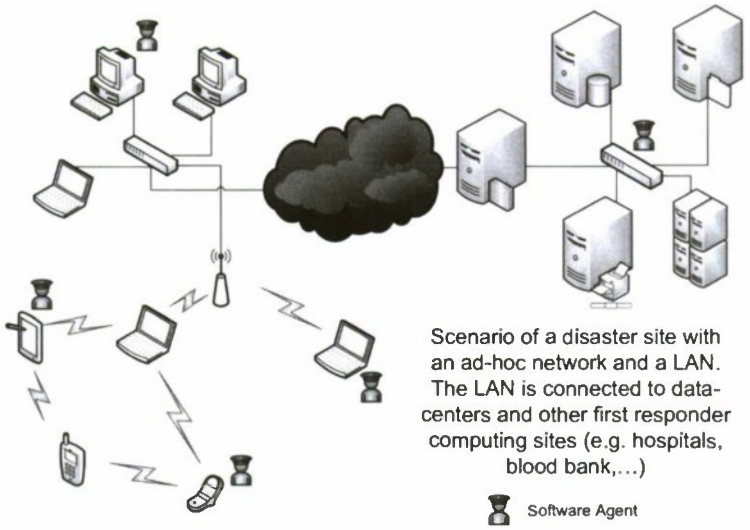

Scenario of a disaster site with
an ad-hoc network and a LAN.
The LAN is connected to data-
centers and other first responder
computing sites (e.g. hospitals,
blood bank,...)
Software Agent

Fig. 1. Example of a disaster response scenario with on-site equipment

To illustrate the applicability of the secure P2P based agent system introduced in this paper, we describe a real life scenario. The object of our scenario is a mine complex with several stakeholders. For crisis operations it is essential that all relevant documentation is made available to the emergency services and on-site personal. One conceivable emergency situation in a mine complex is the collapse of several tunnels. A solution that provides access to all relevant information pertaining to the affected mine(s) such as emergency plans, legal documents, and reports of mining activities, as well as topographical and cartographic material is necessary. By applying the multi-agent system concept using a secure P2P framework as communication layer, it becomes possible to provide a secure decentralized solution to that problem. During an emergency different organizations have to cooperate. This includes fire and rescue, medical, and police, as well as other emergency services such as mine rescue, or utility services. Each service has its own information infrastructure including hard- and software, and employs different security mechanisms. As a common characteristic, we assume that if security measures exist, they rely on cryptographic keys and functions.

2.1 Security Assumptions and Bootstrapping

The security of our solution relies on the secrecy and authenticity of keys stored in nodes. We rely on the following keys to be set up, depending on which security level is used by the node:

- If a ***shared secret key*** is used, we assume a mechanism to set up a secret key for a network with n nodes.

- If ***public-private key pairs*** are used, we assume a mechanism to set up one authentic public-private key pair for each node. In addition, the authentic public keys of legitimate certificate authorities must also be set up for each node.

To set up shared secret keys, most key establishment protocols involve a so-called *trusted third party* or *trusted authority*. Since, we don't want to introduce a single point of failure in our system such protocols are not applicable. We require only one shared secret key instead of pair-wise shared keys which can be efficiently handled by pre-deployed keying. Thus, we can either use a single network-wide key shared by all nodes or a set of keys randomly chosen from a key pool such that two nodes will share one key with a certain probability [13].

To set up public-private keys we use an offline PKI approach since we want to prevent a single point of failure and want to allow node addition during system operation. Thus, the private and public key as well as the trusted authority's public key are embedded in each node. The public keys of other nodes are authenticated using the trusted authority's public key. Using such a system allows us to provide the required authentication but it is not possible to handle revocation. In order to also enable revocation one can either use a distributed PKI solution such as [14, 15] or implement a distributed revocation system [16].

It is necessary to note that our system is not intended as an open system. We don't allow arbitrary nodes to join and thereafter establish secure communication through key agreement since these mechanisms don't provide node authentication in the absence of a trusted authority. Although it is possible that arbitrary nodes join our system, security is only provided for nodes which possess authentic credentials. These legitimate nodes can join and communicate in a secure manner establishing a virtual private overlay.

3 Secure P2P Framework

The secure P2P framework (SePP) is a comprehensive solution for establishing an unstructured P2P network in a secure manner. It provides secure mechanisms for creating and maintaining the overlay network, establishing and managing groups and security administration. The design and implementation of all these mechanisms and protocols is based on a simple but efficient security concept [17]. This security concept has been designed with heterogeneous multi-hop network environments in mind. Another aspect in the design of the security concept was configurability. Thus, giving each node the freedom to select its desired level of security with respect to its capabilities.

3.1 SePP Security Concept

The SePP security concept provides a simple way to select adequate security measures. This allows achieving a specific security level in the face of heterogeneous nodes with diverse capabilities. The features of a security concept are of course important. Especially in our case, the capability to support powerful workstations, as well as constrained mobile devices is equally important. For the remainder of this document we call this scalability. Given our scenario, the security concept must also address node mobility. Nodes with different mobility patterns must be able to participate in the network. Thus, the underlying mechanisms have to cope with a changing environment in a secure and efficient manner. The last characteristic is transparency. Since nodes with different capabilities can participate in the network, the achievable *security level* of a specific communication session must be determinable in advance.

All secure communication mechanisms are considered with the same basic *group* concept in mind. A group is simply a virtual aggregation of an arbitrary amount of nodes which follow the same rules and use the same protocols. Every node can communicate with any other node inside a group. Every virgin node belongs to a *default group* after it has joined SePP. Thereafter, a peer can create or join other groups which are composed of a subset of the peers belonging to the default group.

3.2 Security Levels

There are three different aspects of security which apply to every group:

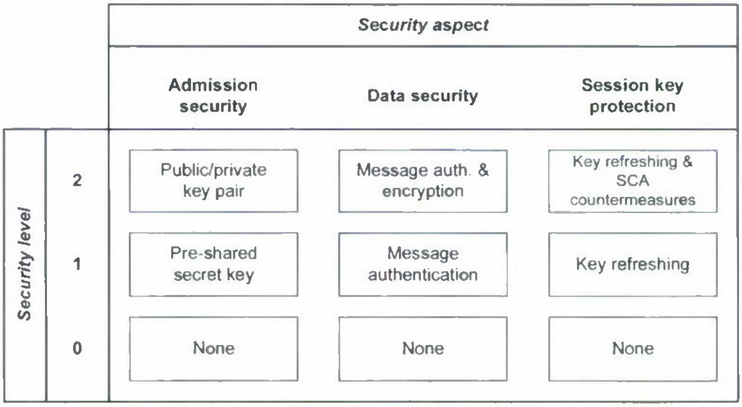
1. *Establishing secure communication (admission security)*
2. *Performing secure communication (data security)*
3. *Upholding secure communication (secret protection)*

Establishing secure communication relates to secure group administration. It combines entity authentication and authorization, secure neighborhood mechanisms and secure bootstrapping. Most notably the join process is addressed with this aspect. After successful authentication each node owns a secret key which is shared amongst the group members. This key is called *session key* and is now used in addition to existing keys in order to increase the performance of performing secure communication between group members. This means for instance that the routing information is protected or all messages are protected group-wise. The benefits of a session key are that it can be updated to protect against side-channel attacks or to exclude misbehaving nodes from the network. Upholding secure communication relates to preventing and limiting damage from exposed session keys. This can be achieved through means such as side-channel attack protection, malicious peer detection or session key refreshing. The overall security for SePP can be set individually on three separate axes. These three axes conform to the three aspects given above. In figure 2 we have outlined the different aspects of security.

For simplicity we only use three different levels of security. These levels are *low (0)*, *medium (1)*, and *high (2)*. Each of these security levels can be chosen differently for every security aspect independently. It is also possible to create more and/or different security levels depending on the system requirements.

Each of these security levels is associated with a specific kind of credential or cryptographic key. For instance, security level *high* requires each node to possess a valid and legitimated public and private key pair. For the *medium* level the nodes must possess a shared secret key for authentication. No secret information is used in security level *low*. Meaning, that if the security level *low* is selected, every node can participate in the network.

In figure 3 nodes are grouped into different security levels. The peers in the light grey area belong to security level *medium*. The peers in the dark grey area belong to security level *high*. All peers outside these areas belong to security level *low*. Our system has been designed in such a way that peers with higher security levels also belong to the lower security levels and are provided with the required credentials for their operation.



		Security aspect		
		Admission security	Data security	Session key protection
Security level	2	Public/private key pair	Message auth. & encryption	Key refreshing & SCA countermeasures
	1	Pre-shared secret key	Message authentication	Key refreshing
	0	None	None	None

Fig. 2. Levels of security in the group concept

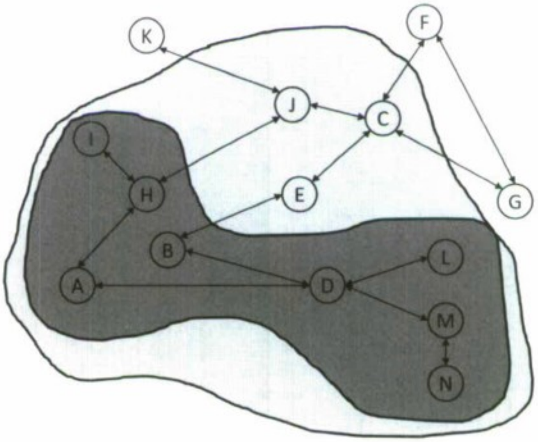


Fig. 3. Overlay network topology with groups of different security levels

4 JADE Multi-Agent System

The Java Agent Development Framework (JADE) is a middleware which simplifies the development of FIPA-compliant agents. It provides support for server and desktop computers and constrained devices. It has been designed under consideration of scalability and supports it throughout the complete development cycle.

JADE commonly uses RMI for the purpose of inter-agent communication. RMI is a Java technique to provide method invocation over a TCP/IP network. The main-goal behind the design of RMI was to easily provide an architecture where code can dynamically be loaded from a server. This is achieved facilitating a client/server architecture and object serialization. A central entity for method binding, called the RMI registry, enables methods to be called from RMI clients remotely even over a network.

The JADE Message Transport Protocol (MTP) default implementation is based on Java RMI. Also, the current version of JADE possesses a well defined interface for implementing other transportation protocols. Unfortunately, this interface's architecture is heavily RMI orientated. Thus, any optional MTP implementation must adapt its messaging system to *emulate* the RMI work-flow.

4.1 JADE Messaging

In order to understand the implementation of the MTP layer, it is first necessary to understand how JADE sends messages. Therefore, we now describe JADE's message transmission sequence. In the following we have to distinguish between the *main peer*, which is responsible for managing the agent platform with its nodes and services and the *remote peer* which uses the main peer's interfaces through proxies. The first step in creating an agent platform is to instantiate the main peer. Thereafter, a new *MTPManager* is created and the *PlatformManager* is used to advertise the JADE Platform. Finally, a *PlatformProxy* is established

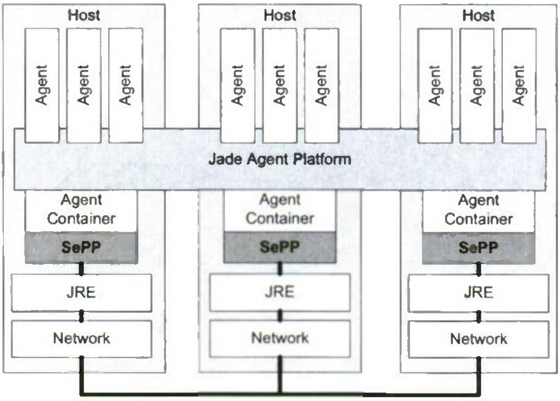


Fig. 4. Secure Jade Architecture

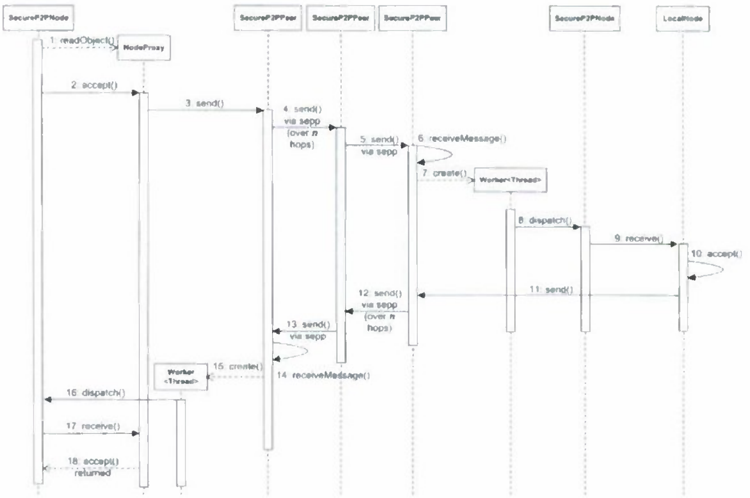


Fig. 5. Message sequence

through the specification of the main peer's address during the creation of the remote peer. With this proxy the remote peer is able to access all the functions available in the platform.

After the remote and main peer are connected we create a JADE node on each peer. Nodes are the agent platform's inter-agent communication mechanism. Each node is associated with an array of services running on the node. If the remote peer wants to send a message to the main peer the remote peer asks the *PlatformManager* via the *PlatformProxy* which services are running on the node of the main peer. During this request, the main peer serializes his local node and the remote peer receives a proxy to the node of the main peer. With this proxy the remote peer is able to access the remote peer's node. From now on, the two agent containers have the ability to communicate with each other. For an example of the JADE messaging process see figure 5.

5 Secure Multi-Agent System

Based on the general overview in the previous section, we now present our implementation of a secure messaging mechanism. For an architectural overview of our implementation see figure 4. The first step was to emulate the RMI behavior on top of SePP using different message classes. A message class is a generalization of a method call. The knowledge about which method should be called and how it is parameterized is encapsuled within the concrete message classes. We require two message classes per method. First, a request message class, whose payload is the parameters of the method. Second a response message class, which contains the return value of the corresponding method. We chose this design, because it allows convenient message dispatching and because this architecture is extendible.

5.1 Implementation

The *SecureP2PIMTPManager* is the general entity that advertises the agent platform and it provides the remote peers with access to the main peer. The functionality of the *SecureP2PIMTPManager* is comparable to the RMI registry as mentioned above, with the difference that we don't have a special storage facility, where the remote methods are registered. The main peer listens for incoming requests from the remote peers.

Every communication between the two peers is done via the *PlatformProxy*. As an example we observe the behavior of the *addNode* method which is called when a new agent container is added. When called from a *PlatformProxy* the *addNode* method creates the appropriate request to the remote *PlatformManager*. This request is sent to the main peer. On the main peer the message is dispatched to the local *PlatformManager* where the request is processed and a new container is created. The return value of the method is sent back to the remote peer. If the *PlatformProxy* is a proxy for a local object, the same request is created. Instead of sending it over the network, it is dispatched locally. JADE also facilitates a different transmission concept which enables sending from a *NodeProxy* to a *LocalNode*. This mechanism is almost identical except for the proxy generation process.

The class *SecureP2PPeer* is the interface to the SePP network through the SePP framework API. This API allows classes which implement the *Component* interface to register themselves to received messages with a specified message type. If a message is received in the *SecureP2PPeer*, the *receivedMessage* method is called. Inside this method, the received byte stream is un-marshalled and a *JADEMessage* is created. This JADEMessage is then passed to the registered *Component*.

Due to the limited space available, we had to omit many details of our implementation.

5.2 A Message Sending Sequence

This section combines the concepts discussed so far and illustrates a sample communication sequence of the JADE platform. We chose he method *accept* for an illustration example of a message sending sequence. *Accept* is used by the agents for communicating within the middleware. We presume that the *PlatformProxy* was already created and the two nodes are ready to send.

The sequence diagram depicted in figure 5 illustrates the whole messaging process, starting from creating the *NodeProxy* until the return of the method *accept*. In the first step the *NodeProxy* is created after serializing the SecureP2PNode from higher layers of the JADE middleware using a mechanism called *transparent proxy generation*. Afterwards the *accept* call on the SecureP2PNode is delegated to its Proxy. This proxy creates a request message and sends it via the SecureP2PPeer class. On the remote container the message is received and a new worker thread is created. This new worker thread forwards the message to its corresponding *JadeComponent*, in this case we have a *LocalNode*. The next step is to invoke the method related to the message and forward the return value via the SecureP2PPeer. Back on the main peer the message is dispatched and the original emphaccept call returns.

6 Security Analysis

The main contribution of our work is to first enable multi-agent systems to work in multi-hop environments and second to provide means to do that in a secure manner. To establish the exact boundaries of our security analysis we first define our assumptions on the environment.

1. Each peer or user that is a part of the multi-hop enabled communication subsystem must be authenticated. This means, each user joining the system must provide a prove of it's identity depending on the security requirements of the overall system. This can involve either something the user knows or something the user has.
2. A multi-hop enabled MAS must provide means to communicate even in the absence of direct connections between the different parts of the MAS. Thus, it must be possible for agents to communicate with each other in a hop-by-hop manner in addition to direct communication. This fact requires secure routing algorithms which guarantee that only legitimated and trusted hops are used to forward data. Otherwise, malicious peers can disrupt communications or separate parts of the MAS at their will.
3. The data communication of the MAS must be protected. Depending on the system security requirements this can include protection from fabrication, modification, interruption and interception. This translates to data authentication and data confidentiality in addition to reply protection.
4. Participating users are considered trustworthy. Thus, only attacks from external entities are considered (outsider attacks). In addition, it should be possible to identify attacks from malicious insiders if the security requirements for such an identification are met.

To conclude the list of assumptions it has to be remarked that it is always important to scale the security mechanisms according to the potential damage. Every user should use the best available security measures given his resources.

6.1 Analysis of the SePP-Jade Solution

The join process in SePP is secured and provides authentication. This guarantees that only legitimated peers can join and participate in the overlay network. After peers have joined the SePP network, different secure routing algorithms can be used to guarantee the integrity of the network. These routing protocols allow for a secure establishment and maintenance of end-to-end paths in the overlay network. Depending on the overall security requirements and selected security level, SePP provides means for data authentication and confidentiality. SePP uses well known cryptographic algorithms and protocols to ensure the security of the data. Reply protection is also integrated into SePP through its messaging system.

The security levels and SePP are designed in such a manner that for *low* everybody can participate and no cryptographic protection is applied. Thus, no guarantees on the security of the system can be given.

In security level *medium* shared secret keys are used for node authentication. Therefore, the secure routing protocol which enforces authentication based on these secret keys provides protection from outsider attacks. In this security level insider attacks are still possible.

In security level *high* digital signatures are used to secure the established routes. Thus, also insider attacks from non-collaborating peers can be prevented. Non-collaboration means that the peer does not control any other peer on a path from the source to the destination. If peers collaborate they can always perform a wormhole attack by tunneling the route request from one peer to the other and effectively shorten the route length. This attack only works if the controlled path is also the fastest. Otherwise the request would arrive to late to be selected from the destination as new route to the source. For instance, peer D is the source and peer C is the peer who meets the destination information requirements. Now if peer B and peer J of figure 3 collaborate and the tunneled request would arrive earlier than the other one traveling over A and E, this attack still succeeds. Such attacks can currently only be mitigated if location information and synchronized clocks are used [12,18].

7 Performance Evaluation

To benchmark our implementation we compared it with JADE's *out-of-the-box* RMI implementation. We used the *PartyAgent* application from the JADE examples. Within this application we created 500 *PartyGuest* agents which send several messages to each other in order to pass on some gossip. We measured the time from the start of the application until all messages are sent, and the party has officially ended. During this time about 7000 messages have been sent and received from the agents. The party host agent is responsible for about 99% of the messages.

The tests have been performed on HP personal computers with Intel Core 2 Duo E8600 processors with 3.33 GHz and 4 GB RAM and Windows 7 as operating system. The SePP framework implementation has been executed on Java JDK 6 Update 17 runtime environments. During the test no virus program or firewall was active.

The values in table 1 have been obtained from different runs of the *PartyAgent* application. These values show how long one specific run of the application took. In the first column the values from the vanilla JADE version using RMI without any security feature are presented. The next two columns show the amount of time it took for the same application to finish using SePP with security level *medium* as communication layer. The first one has been obtained for the case that the two peers have a direct connection. The second one depicts the case

Table 1. Processing time of the different security levels at the participating peers

	RMI [s]	SePP (direct) [s]	SePP (1 hop) [s]
1	4.40	15.40	16.20
2	3.70	14.80	16.90
3	3.40	14.30	15.90
4	3.60	16.10	15.80
5	3.50	14.00	15.40
6	3.90	14.60	16.10
7	3.40	13.40	15.40
8	3.70	15.00	15.80
9	3.30	16.00	15.30
10	3.40	14.30	16.00
Mean	3.63	14.75	15.88

that there is one intermediate hop between the main peer and the remote peer. The run time of the JADE version using SePP is about four to five times slower than the RMI version. This fact can be attributed to increased processing time for cryptography and the P2P management and communication overhead. The usual round trip time in SePP without transmission latency is about $500\mu s$. Thus, sending and receiving 7000 messages alone would account for 3.5 seconds, which already is the mean run time of the RMI version.

8 Related Work

Multi-agent systems have been used in disaster response scenarios previously. For instance disaster response [4], [5] have shown that MAS can be quite helpful under such circumstances. But these approaches haven't addressed security or multi-hop communication requirements in anyway. They where only concerned with showing the features MAS can provide in disaster response.

The JADE developers itself have proposed a security extension for its framework [19]. Anyhow, this security framework is only intended as add-on for JADE and therefore doesn't address all requirements for security in such challenging environments. Also their extension only provided interfaces for JAAS (Java Authentication and Authorization Service) and they didn't implement any security itself or give instructions on how to use it. There exist also some other works which have addressed security in JADE. But they are all theoretical and only outline the requirements and discuss the necessary security features formally. One such effort is [20].

Some other works have also addressed security of multi-agent systems. But almost all have only been of theoretical nature. They have outlined the requirements in terms of security and shown what attacks and threats are possible with in the domain of MAS. The most prominent such work is [6].

9 Conclusion

In conclusion, the proposed multi-agent system with SePP as underlying communication infrastructure enables the use of agent technology in multi-hop environments in a secure way. We provided simple means of integrating and enhancing existing MAS with secure communication mechanisms without the need for redesign or re-implementation of the MAS itself. We introduced an RMI-style interaction layer which mediates between the MAS on top of a secure P2P framework. The security management is separated from the MAS application and can be adjusted according to the needs of the participating entities. With our solution it is possible to comply with various security requirements in a fine grained manner since it is possible to select security levels from a global to a group scale. The introduced security guarantees increased robustness and the added multi-hop functionalities justify the marginal negative impact on communication performance compared to JADE's RMI solution. Furthermore, we believe that our solution has the potential to increase the efficiency of emergency response operations for scenarios where an existing network infrastructure has been destroyed or disrupted and the different parties had to rely on proprietary or fall-back communication facilities.

References

1. van Dyke Parunak, H., Brueckner, S., Sauter, J.: Digital pheromone mechanisms for coordination of unmanned vehicles. In: AAMAS 2002: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 449–450. ACM, New York (2002)
2. Maturana, F.P., Staron, R.J., Hall, K.H.: Methodologies and tools for intelligent agents in distributed control. IEEE Intelligent Systems 20(1), 42–49 (2005)
3. Picard, G., Bernon, C., Gleizes, M.P.: Etto: Emergent timetabling by cooperative self-organization. In: Brueckner, S.A., Di Marzo Serugendo, G., Hales, D., Zambonelli, F. (eds.) ESOA 2005. LNCS (LNAI), vol. 3910, pp. 31–45. Springer, Heidelberg (2006)
4. George, J.P., Gleizes, M.P., Glize, P., Regis, C.: Real-time simulation for flood forecast: an adaptive multi-agent system staff. In: AISB 2003: Proceedings of the 3rd Symposium on Adaptive Agents and Multi-Agent Systems, pp. 7–11 (2003)
5. Schurr, N., Marecki, J., Tambe, M., Scerri, P.: The future of disaster response: Humans working with multiagent teams using defacto. In: AAAI Spring Symposium on AI Technologies for Homeland Security (2005)
6. Jansen, W.A.: Countermeasures for mobile agent security. Computer Communications 23(17), 1667–1676 (2000)
7. Wallach, D.S.: A survey of peer-to-peer security issues. In: International Symposium on Software Security, pp. 42–57 (2002)
8. Campadello, S.: Peer-to-peer security in mobile devices: A user perspective. In: P2P 2004: Proceedings of the Fourth International Conference on Peer-to-Peer Computing, pp. 252–257. IEEE Computer Society, Los Alamitos (2004)
9. Mondal, A., Kitsuregawa, M.: Privacy, security and trust in p2p environments: A perspective. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 682–686. Springer, Heidelberg (2006)

10. Lundberg, J.: Routing security in ad hoc networks. Technical Report 501, Helsinki University of Technology (2000)
11. Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
12. Hu, Y.C., Perrig, A., Johnson, D.B.: Packet leashes: a defense against wormhole attacks in wireless networks. In: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1976–1986. IEEE, Los Alamitos (2003)
13. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: CCS 2002: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 41–47. ACM, New York (2002)
14. Aberer, K., Datta, A., Hauswirth, M.: A decentralized public key infrastructure for customer-to-customer e-commerce. International Journal of Business Process Integration and Management 1(1), 26–33 (2005)
15. Lesueur, F., Me, L., Tong, V.: An efficient distributed PKI for structured P2P networks. In: P2P 2009. IEEE Ninth International Conference on Peer-to-Peer Computing, September 2009, pp. 1–10 (2009)
16. Cholez, T., Chrisment, I., Festor, O.: A distributed and adaptive revocation mechanism for p2p networks. In: ICN 2008: Proceedings of the Seventh International Conference on Networking, pp. 290–295. IEEE Computer Society, Los Alamitos (2008)
17. Kraxberger, S., Payer, U.: Security concept for peer-to-peer systems. In: IWCMC 2009: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing, pp. 931–936. ACM, New York (2009)
18. Poovendran, R., Lazos, L.: A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. Wirel. Netw. 13(1), 27–59 (2007)
19. JADE Board: Jade security add-on guide. Technical report, TILAB S.p.A (2005)
20. Vila, X., Schuster, A., Riera, A.: Security for a multi-agent system based on jade. Computers and Security 26(5), 391–400 (2007)

In the Track of the Agent Protection: A Solution Based on Cryptographic Hardware

Antonio Muñoz, Antonio Maña, and Pablo Antón

Computer Science Department
University of Málaga, Spain
{amunoz,amg,panton}@lcc.uma.es

Abstract. The agent-based computing represents a promising paradigm for emerging ubiquitous computing and ambient intelligence scenarios due to the nature of the mobile agents that fit perfectly in these environments. However, the lack of the appropriate security mechanisms is hindering the application of this paradigm in real world applications. The protection of malicious hosts is the most difficult security problem to solve in mobile agent systems. In this paper we describe our solution, which is a mechanism to solve this problem. Our work is based in a new agent migration protocol based on the use of tamper resistant cryptographic hardware. Concretely, we base our work on the use of the Trusted Computing technology. The result of our work is a library built on JADE that implements the secure migration for agents named Secure Migration Library for Agents (SecMiLiA). This library provides a friendly use of the Trusted Computing technology for agent based system developers.

Keywords: TPM, cryptographic hardware, agent protection.

1 Introduction

While mobile agent paradigm expresses many advantages over the traditional network computing models [4], the code mobility of the mobile agents brings some severe security problems. Current research efforts in the security of mobile agent field adopt two different points of view. Firstly, from the platform perspective, we need to protect the host from malicious mobile agents such as viruses and Trojan horses that are visiting it and wasting resources. Secondly, from the mobile agent point of view, we need to protect the agent from malicious hosts. Both points of view have attracted much research effort. In [7] authors show that scientific community put many efforts in this field, indeed many applications exist based on this technology. However all these efforts loose their values due to they are not based on a secure robust basis to build applications. This fact encouraged us to afford the task of the malicious hosts. For this purpose we make use of a tamper resistant cryptographic hardware. Because of the recent approaches in the Trusted Computing technology and their cheaper price we choose this technology to implement our protocol. However, the secure migration protocol can be implemented on any tamper resistant cryptographic

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 284–297, 2010.

hardware, such as the smartcards. In [7] two mechanisms are presented to provide security for the agent based systems. Firstly, a software based solution built on the protected computing approach [6] is introduced. Secondly, a complete description of a hardware based approach is explained. This approach is the basis of the work presented in this paper. This paper is organized as follows. Section 2 deals with the State of the Art. Section 3 gives a perspective of the role of the Trusted Computing in the agent protection. Section 4 presents the SecMiLiA. Section 5 describes some issues found in the design and development of the library. Section 6 gives an overview of the main services provided by the library. Section 7 introduces some supporting technology. Finally section 8 gives some concluding remarks.

2 State of the Art

Some mechanisms are oriented to the protection of the host against malicious agents. Among these, SandBoxing [9], proof-carrying code [8], and a variant of this technique, called proof-referencing code [2]. One of the most important problems of these techniques is the difficulty of identifying which operations (or sequences of them) can be permitted without compromising the local security policy. Other mechanisms are oriented towards protecting agents against malicious servers. Among them the concept of sanctuaries [12] was proposed. Several techniques can be applied to an agent in order to verify self-integrity and avoid that the code or the data of the agent is inadvertently manipulated. Anti-tamper techniques, such as encryption, checksumming, anti-debugging, anti-emulation and some others [1,11] share the same goal, but they are also oriented towards the prevention of the analysis of the function that the agent implements. Additionally, some protection schemes are based on self-modifying code, and code obfuscation [3]. The technique known as co-operating agent [10,5] consists on the distribution of critical tasks of a single mobile agent between two co-operating agents. Each of these agents executes the tasks in one of two disjoint sets of platforms. Finally there are techniques that create a two-way protection. Some of these are based on the protected computing approach [7]. Most of these approaches are software based solutions. However, it is important to consider the fact that the degree of confidence in software-only security solutions depends on several factors sometimes uncontrollable, such as their correct installation and execution. This can be affected by all other software that has been executed on the same platform. For this reason, experts conclude that trusted hardware is needed as the basis for security solutions. We aimed that our solution takes advantage of the recent advances in the trusted computing technology.

Agent migration consists on a mechanism to continue the execution of an agent on another location. This process includes the transport of agent code, execution state and data of the agent. The migration in an agent based system is initiated on behalf of the agent and not by the system. The main motivation for this migration is to move the computation to a data server or a communication partner in order to reduce network load by accessing a data server a communication partner by local communication. Then migration is done from a source agency where agent is running to a destination agency. Migration can be performed by two different ways. Moving is the process in which the agent is removed from the source agency when is copied in the destination agency. Cloning consists on the agent is copied to the destination agency. Henceforth, the two copies of the agent coexist executing in different places. In the remainder of this paper and at least stated explicitly we will use the term migration to refer to both cloning and moving of agents. Our approach is addressed on achieve a secure migration process. For this reason we propose a hardware-based mechanism to provide security to agent systems. The TPM provides mechanisms, such as cryptographic algorithms, secure key storage and remote attestation that provides important tools to achieve a high level of security.

3 The Trusted Computing Technology in the Agent Protection

Previously we aimed that is essential the integration of new trusted security mechanisms in the agent software field to achieve a reasonable security level. For this reason we propose a proposal based on the appeals of the tamper resistant cryptographic hardware. This kind of technology provides some mechanisms, such as cryptographic algorithms, secure key storage and remote attestation that are essential to achieve a high level of security. The main appeal of the SecMiLiA is that agent software developers are liberated of security engineering related tasks, due to the underlying security of our approach.

However our main objective is to provide a high level of security for agent execution avoiding possible attacks of the hosts. Then, we propose the use of the trusted computing technology; indeed we use the Trusted Platform Module (TPM). Mainly, due to TPM is compliant with the security requirements mentioned above and this technology has other features like the standardisation and the growing integration of this technology in the market. Additionally, the supporting provided by many important companies leaders in the IT security sector is a considerable appeal of this technology. TPM is the cornerstone of our approach due to the security of our system is relies on it, this is following explained. We identified two main pillars of agent protection. Firstly we have to protect the execution element. The protection of this element is provided by the root of trust provided of the TPM. It is based on controlling that only a restricted set of operations can be executed. Secondly we have to protect the migration procedure, for this purpose, we use the remote attestation functionality provided by the TPM. In order to facilitate the use of this mechanism we developed a full library based on the most used agent platform JADE.(Java Agent DEvelopment Framework), which is a software Framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications [14] and through a set of graphical tools that supports the debugging and deployment phases.

The basic idea behind the concept of Trusted Computing is the creation of a chain of trust between all elements in the computing system, starting from the most basic ones. Consequently, platform boot processes are modified to allow the TPM to measure each of the components in the system and securely store the results of the measurements in Platform Configuration Registers (PCR) within the TPM. This mechanism is used to extend the root of trust to the different elements in the computing platform. Therefore, the chain of trust starts with the mentioned above TPM, which analyses whether the BIOS of the computer is trusted and, in that case, passes control to it. This process is repeated for the master boot record, the OS loader, the OS, the hardware devices and finally the applications. In a Trusted Computing scenario a trusted application runs exclusively on top of trusted and pre-approved supporting software and hardware. Additionally the TC technology provides mechanisms for the measurement (obtaining a cryptographic hash) of the configuration of remote platforms. If this configuration is altered or modified, a new hash value must be generated and sent to the requester in a certificate. These certificates attest the current state of the remote platform. We have seen that several mechanisms for secure execution of agents have been proposed in the literature with the objective of securing the execution of agents. Most of these mechanisms are designed to provide some type of protection or some specific security property. Despite they only provide partial solutions to the agent systems security.

We introduce the case that an agent executing in an agency (source agency) plans to migrate to a different agency (destination agency). Both agencies take measures of some system parameters, which determine the security, for instance BIOS, keys modules from Operating System, active processes and services in the system. Through these parameters an estimation of the secure state of the agency can be done. Values taken are securely stored in the trusted device, in such a way that cannot be either access or modified unauthorized. Agency has the ability to report configuration values previously stored to other agencies in such a way that these can determine its secnrity. Before the migration the agent requests to the source agency to determine the trustworthy on destination agency. By means of this process an agent in a secure agency can extend the limit of its confidence to other agency once the security of destination agency is tested.

4 Our Final Result: The Secure Migration Library (SecMiLiA)

In this section we introduce the Secure Migration Library (SecMiLiA). This library provides the secure migration functionality. In order to give a friendly use of the security mechanism provided. We aimed that SecMiLiA is based on the JADE platform. The main reasons for that fact are following described. Firstly, because of the widespread use of the JADE platform in the agent community; and secondly because of the interplatform migration mechanism provided by JADE. The figure 1 depicts a block diagram that shows how the SecMiLiA is built on JADE.

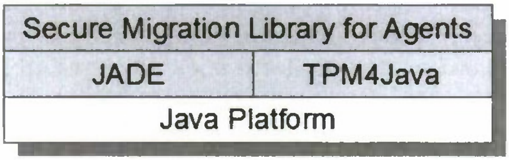


Fig. 1. block diagram

4.1 The Set of Minimum Requirements of Our Library

The most relevant objective in the design of this library is the provision of a secure environment in which agents can be securely executed and migrated. It is relevant the easy integration in JADE, that is, no modifications in JADE might be done, but as secondary aspect. Similarly to the provision of a friendly use for agent software developers, who are not security expert. And the provision of a library that complains with the existing security solutions.

As a final result we obtained a library that provides security to software agents on JADE. We achieve this security level by means of a mechanism that allows the secure migration. This secure mechanism is based on the testing the trust of destination agency before the migration process actually is performed that we explain in the next section. This guarantees that agent execution is always performed in a secure environment. This gives a solution to the problem of the malicious hosts. Thus, agent reaches a secure environment where its execution goes on, in such a way that agents cannot modify the host agency.

We identified some minimum requirements in the design process of the library. These requirements were grouped in two different sets, functional and non-functional requirements. On the one hand, concerning the functional requirements, the library must provide a mechanism for secure agent migration in JADE platform, in such a way that the agent can extend its trustworthy limits by means of adding secure agencies, both from its platform and from remote platforms. It is important to mention the fact that, each agency must provide local functionality, which is allowing an agent to migrate to a destination platform. Similarly, each agency must provide the functionality to allow to other agencies take integrity measures to determine if its configuration is secure. The library must implement the protocol to allow configuration related information to interchange from an agency to a different one. This is required to be implemented in such a way that both agencies are trusted from the origin of this information. Last but not least concerning functional requirements the library must use trusted hardware. In this case we used a TPM to stored securely agencies data integrity and reporting data to the agencies, which requested. On the other hand, related with the non-functional requirements, we believe that the library might be integrated in the JADE platform, in such a way that the library use does not imply modifications in the JADE configuration. As well as, the operation of the library must be transparent to the user. Library must ease the adaptation to existent solutions to use security mechanisms provided in such a way that the number of modifications is reduced at maximum. A generic security mechanism is provided to be easily adapted to concrete solutions. And it is important that the library allows the possibility to be easily extended with future improvements and new functionalities.

4.2 Secure Migration Protocol

Following we include an overview of the protocol that is the basis of our library. This protocol is the basis to provide the security to the migration. We analyze the different attestation protocols as well as the secure migration protocol. Then we study their benefits to design the secure migration protocol. In [7] authors described a draft of this protocol, which provides some key ideas to take into account during the design process of the final protocol. Let us assume that the agent is executing in a secure platform. Thus, the agent trusts in this platform to check the migration security. Additionally, it is interesting to mention the necessity of the use of the TPM to obtain and report configuration data. More relevant ideas provided are that a protocol shows how an agent from the agency requests to TPM the signed values from PCRs. Besides, the protocol shows how the agent obtains platform credentials, these credentials together with PCRs signed values allow to determine whether the destination configuration is secure.

Finally we analyse in depth the protocol .More relevant ideas from this protocol are; the use of an attestation identity key (AIK) to sign the PCR values; the use of a certification authority (CA) that validates the attestation identity key (AIK); and the use of configurations to compare received results from remote agency. We designed a new protocol based on the study of the trusted computing technology. Our protocol has some characteristics, for instance; the agency provides to the agent the capacity to migrate by a secure way; and the agency uses a trusted platform module that provides configuration values stored in PCRs. The trusted platform module signs PCRs values using a specific attestation identity key for the destination agency; in such a way that data receiver knows securely the TPM identity, which is signed. A Certification Authority generates the needed credentials to verify the AIK identity. Together with signed PCRs values the agency provides attestation identity key credentials producing the signature. This signature is used to verify that the data are exactly from the source TPM. A further description of this protocol is included in [15].

4.3 Verification of Secure Migration Protocol with AVISPA

The secure migration protocol described above is the basis of this research. Thus, we want to build a robust solution, for this purpose the next step is validation of this protocol. Among different alternatives we selected a model checking tool called AVISPA.

AVISPA is an automatic push-button formal validation tool for Internet security protocols, developed in a project sponsored by the European Union. It encompasses all security protocols in the first five OSI layers for more than twenty security services and mechanisms. Furthermore this tool covers (that is verifiable by it) more than 85 of IETF security specifications. AVISPA library available on-line has in it verified with code about hundred problems derived from more than two dozen security protocols. AVISPA uses a High Level Protocol Specification Language (HLPSL) to feed a protocol in it; HLPSL is an extremely expressive and intuitive language to model a protocol for AVISPA. The operational semantic is based on the work of Lamport on Temporal logic of Actions. Communication using HLPSL is always synchronous. Once a protocol is fed in AVISPA and modelled in HLPSL, it is translated into Intermediate Format (IF). IF is an intermediate step where re-write rules are applied in order to further process a given protocol by back-end analyzer tools. A protocol, written in IF, is executed over a finite number of iterations, or entirely if no loop is involved. Eventually, either an attack is found, or the protocol is considered safe over the given number of sessions.

System behaviour in HLPSL is modelled as a "state". Each state has variables which are responsible for the state transitions; that is, when variables change, a state takes a new form. The communicating entities are called "roles" which own variables. These variables can be local or global. Apart from initiator and receiver, environment and session of protocol execution are also roles in HLPSL. Roles can be basic or composed depending on if they are constituent of one agent or more. Each honest participant or principal has one role. It can be parallel, sequential or composite. All communication between roles and the intruder are synchronous. Communication channels are also represented by the variables carrying different properties of a particular environment. The language used in AVISPA is very expressive allowing great flexibility to express fine details. This makes it a bit more complex than Hermes to convert a protocol into HLPSL. Further, defining implementation environment of the protocol and user-defined intrusion model may increase the complexity. Results in AVISPA are detailed and explicitly given with reachable number of states. Therefore regarding result interpretation, AVISPA requires no expertise or skills in mathematics contrary to other tools like HERMES[13] where a great deal of experience is at least necessary to get meaningful conclusions.

Of the four available AVISPA Back-Ends we chose the OFMC Model, which is the unique that uses fresh values to generate nonce's. However, this alternative requires a limit value for the search. The results of our research are the following:

```
SUMMARY  SAFE
....
STATISTICS
  parseTime: 0.00s
  searchTime: 564.34s
  visitedNodes: 18 nodes
  depth: 2000 plies
environment()
```

These results show that the summary of the protocol validation is safe. Also some statistics are shown among them depth line indicates 2000 plies, but this process has been performed for 200, 250, 300, 400, 500 and 1000 of depth values with similar results. A further description of this validation is out of the scope of this paper.

5 Design and Deployment of the Library

In the development process were found some issues, such as a JADE system is composed for a platform that keeps a main container in which agents are deployed. Additional containers can be added to this platform, some of them can be remote containers, and different platform can interacts among them, allowing the migration of the agents between the agencies. Henceforth, we consider the same platform and agency. Taking into account the JADE structure, we conclude that two different kinds of migration exists, migration among containers from different platforms and migration from containers in the same platform. In the case that the migration is from containers from different platforms, the agent migrates from a container from source agency to the destination agency main container. In such a case that destination agency is not a JADE built-on platform the architecture can be different, depending on the platform. In the other case, the agent migrates from a container to another one but in the same platform. Both migration processes imply some security concerns. The platform migration is not secure because the main container from the source platform can be untrusted, the migration between containers has the same problem, it is, if destination container is not trusted; and the migration is not secure. Secure migration library solves both risen problems. In this section we analyse the deployment and the design of SecMiLiA. Firstly, we study the architecture of the library; secondly we show the components and their related functionalities. The main use case is a user that uses SecMiLiA to develop a secure agent based system. We consider a relevant aspect to consider that the user is not a security expert. Traditionally in these kinds of systems, the user defines the set of agents that compound the system. Concretely JADE defines an agent by means of a class that inherits from Agent class, using this new class the agent created is provided of the basic behaviour of an agent. Therefore the user defines the specific behaviour of this agent. Among the most relevant functionalities of a JADE agent we highlight the compatibility with inter-containers migration. Concerning the main migration methods we highlight, doMove (Location1) moves the agent from a source container to a destination one. The method named doClone(Location1, String newName) clones the agent in container1 using newName as the name. Two main services are provided by SecMiLiA. The AgentMobility service performs a secure inter-platform migration in the same platform, and the SecureInterPlatformMobility service, which uses the InterPlatformMobility service to perform the secure intra-platform migration. We mentioned above that JADE Agent class provides two "non-secure" migration methods, for this reason we have created a new class that inherits from this class and redefines migration methods to perform a secure process. This allows a complete integration in the JADE platform, as well as provides a friendly use for agent software developers, who only need instance the SecureAgent class and invoke the secureMigration method.

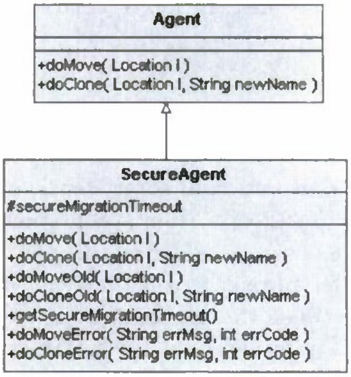

Agent
+doMove(Location l)
+doClone(Location l, String newName)
SecureAgent
#secureMigrationTimeout
+doMove(Location l)
+doClone(Location l, String newName)
+doMoveOld(Location l)
+doCloneOld(Location l, String newName)
+getSecureMigrationTimeout()
+doMoveError(String errMsg, int errCode)
+doCloneError(String errMsg, int errCode)

Fig. 2. SecureAgent Class

6 Main Functionalities of SecMiLiA

This section describes the main services provided by the SecMiLiA as well as the more relevant classes and methods of this library.

6.1 SecureAgentMobility Service

The SecureAgentMobility service provides the secure migration functionality between different containers in the same platform. Concretely, the "Helper" class provides two important methods: (i) "secureMove" that allows secure agents moving securely to destination container and (ii) "secureClone" that provides a secure way to cloning agents in destination containers. The following algorithm presents the case where an agent request for a service to move to a container (c2). In this case the steps are: This protocol considers the service like a unique entity. However, several components belonging to that service are avoided in order to clarify. Other important issue is that the service invokes "doMoveOld" method to start the migration, which functionality is similar to "doMove" from Agent class, moving agent to destination. Previously to the migration, the service checks that destination is secure. This fact allows a similar behaviour of the "doMove" method from "SecureAgent" and the "doMove" method from the "Agent" class. The content of these messages is encapsulated using:"AttestRequest_Interface", and the "AttestData_Interface" interfaces. The "AttestRequest_Interface" provides access to data from a request attestation message, and the "AttestData

Algorithm 1. The agent SA is moved to the container C2

```
SA agent requests for S1 service to move to C2
Service sends a remote attestation request to S2 service.
S2 service accepts the request
S1 service sends requested information to S2.
S2 service responses to S1 sending the attestation result.
S1 service starts agent migration to C2 container.
```

_Interface" provides access to the attestation information from a concrete container. Both interfaces encapsulate information from attestation protocol messages. To continue the attestation procedure the source container completes the needed data using the "set" method. The "secureAgentMobility" service uses the "AttestTool_Implement" class to complete data messages.

The "AttestTool_Implement" class manages attestation protocol messages, that is, the generation of the messages in the sources and the verification in the destination. "AttestTool_Implement" class is provided by access to system TPM, for that purpose the "TPM_Interface" and the "CA_Interface" interface are accessed. This fact allows using both entity functionalities to complete messages. "AttestTool_Implement" class allows to access to system configuration trough the "AttestConfig_Interface" interface. "AttestTool_Implement" class manages TPM access in such a way that attestation protocol can be performed. The "AttestTool_Implement" class behaviour is similar to Key Cache Manager for dealing with the TPM keys. The "AttestTool_Implement" class implements "AttestTool_Interface" interface where secure migration process states codes are defined. These error codes allow to agents to determine the results when doMoveError and doCloneError methods are called.

In order to generate and verify the attestation messages contents is needed the use of a TPM and a certification authority (CA). More relevant reasons for this are that (i) the TPM provides the functionalities to generate attestation data; (ii) the needed functionalities to generate the attestation identity keys (AIK); (iii) the functions to produce the data signature; (iv) and the functions to allow the generation of random nonce values. We have to consider that the Certification Authority (CA) provides the credentials generation. Firstly, the "TPM_Interface" interface provides access to TPM functionalities. Among them we found, the initialization of the interface with TPM module; the generation, dropping and activation of an attestation identity key request from "reqData". The certification authority received the "reqData" to provide the certificates. As well as, the functions to attest the configuration, etc. Secondly, the "CA_Interface" interface provides the functionality to deal with the Certification Authority. This interface contains some functions that provide the certification authority label value, the identification and the public key, as well as the functions to generate the credentials for the attestation identity key. Thirdly, the "AttestConfig_Interface" provides access to platform configuration values, as well as own platform values. This interfaee provides the PCRs indexes to the remote containers, TPM owner password, storage of keys, etc.

A relevant aspect using the attestation identity key (AIK) in the protocol is the production of the signature. The TPM generates the AIK and this must be certified by a valid certification authority. Following we describe how the key is generated and certified.

"AttestTool_Implement" is requested for generate an attestation identity key key and a credentials request to TPM. Then, TPM is used to generate the attestation identity key as well as the request and these are delivered to AttestTool. Request is encrypted in such a way that only the certification authority is able to read. The "AttestTool_Implement" sends a certification authority request. Next the Certification Authority (CA) is used to decrypt the request and generates the credentials which are delivered to "AttestTool_Implement" in such a way that only TPM can decrypt. The "AttestTool_Implement" sends the Certification Authority (CA) responses to TPM. And TPM functionalities are used to decrypt the requested data and sends the key data to the "AttestTool_Implement".

Several interfaces interact in this process; (i) The "AIKRequestData_Interface" contains the needed data to allow the TPM to generate the attestation identity keys (AIK), as well as, to create the credentials generation request. (ii) The "AIKRequestData_Interface" contains the needed data to allow the certification authority (CA) to generate the credentials for the attestation identity keys (AIK). (iii)And the "AIKResponse_Interface" that contains the data to allow the TPM to obtain the credentials generated by the certification authority.

At this point we briefly described some relevant classes. "AIKRequestData _Interface" provides "getIdentityLabel()" method, this returns the attestation identity key label. The "AIKRequest_Interface" provides access to the identity label, data to the certification authority to certify the attestation identity key, AIK public key from TPM, key wrappers, etc. "AIKResponse_Interface" interface provides the functionalities to manage the attestation identity key, which is to get the key handler in TPM, the attestation identity public key, attestation identity key credentials, key wrappers, etc.

Once the attestation identity key reaches the destination the service generates the configuration attestation data, then the signature is produced with these data. Finally, we describe other important elements in this solution, we aim to the credentials. Some classes are dedicated to deal with the credentials. The certification authority generates the attestation identity key credentials defined by "AIKCredentials_Interface" interface.

6.2 SecureInterPlatformMobility Service

The SecureInterPlatformMobility service uses most of used elements in the SecureAgentMobility service. However, in this case we deal with migration

Algorithm 2. Secure migration protocol

```
SA agent requests S1 service to move to C2 container.
S1 service sends a remote attestation request to S1M service from main container of its platform.
S1M service sends a request for remote attestation to source platform AMS, A1.
A1 sends a request for attestation to destination platform AMS, A2.
A2 accepts the request and notifies to A1.
A1 notifies S1M service acceptation.
S1M service notifies S1 service acceptation.
S1 service sends request data to S1M service.
S1M service sends data to A1 request.
A1 sends request data to A2.
A2 responses to A1 sendind the attestation result.
A1 sends result to S1M service.
S1M service sends received result to S1 service.
S1 service starts agent migration to destination platform main container.
```

between different platforms, that is, the service messages between the source container and the destination container are not allowed. This implies that both containers must belong to the same platform. This fact restricts the communication by using "Agent Communication Language" ACL messages. The secure migration protocol for SecureInterPlatformMobility service is following described.

The source container service needs the interaction of the main container service to interact with the destination platform AMS, in such a way that the only way to access to agent management system (AMS) class implemented from the main container. The communication between the source platform AMS and the destination platform agent management system (AMS) is done by agent communication language (ACL) messages. Destination AMS uses "AttestTool_Implement" class to deal with service messages. The rest of service operation component is similar to the SecureAgentMobility afore detailed.

7 Application of Secure Agents to Clouds Computing

The term "cloud" is used as a metaphor for the Internet, based on the cloud drawing used in the past to represent the telephone network and later to depict the Internet in computer network diagrams as an abstraction of the underlying infrastructure it represents. Typical cloud computing providers deliver common business applications online which are accessed from another web service or software like a web browser, while the software and data are stored on servers. Most cloud computing infrastructure consists of reliable services delivered through data centers and built on servers. Clouds often appear as single points of access for all consumers' computing needs.

In general, cloud computing customers do not own the physical infrastructure, instead avoiding capital expenditure by renting usage from a third-party provider. They consume resources as a service and pay only for resources that they use. Many cloud-computing offerings employ the utility computing model, which is analogous to how traditional utility services (such as electricity) are consumed, whereas others bill on a subscription basis. Sharing "perishable and intangible" computing power among multiple tenants can improve utilization rates, as servers are not unnecessarily left idle (which can reduce costs significantly while increasing the speed of application development). A side-effect of this approach is that overall computer usage rises dramatically, as customers do not have to engineer for peak load limits. In addition, "increased high-speed bandwidth" makes it possible to receive the same response times from centralized infrastructure at other site. Obviously the most relevant issue to face is the lack of the appropriate security mechanisms.

However we advocate for a new mode of clouds computing in which not only data are processed by the clouds, instead of it user code´s could be executed in the cloud. Evidently, the security is the cornerstone for the successful of this approach. We envisage a parallelism between the security in this new vision of clouds computing and mobile agent systems, there are pieces of software that are executed in different environments. Both environments present similar security issues, and we propose to use the same model presented in this paper to clouds computing.

8 Conclusions and Future Research

In this paper we provide a general solution based on solving the problem of the "malicious hosts". Our approach is based on the "Trusted Computing Module" security capabilities. Despite our solution is a friendly library, but this is built on a robust secure basis as we explained in this paper. Possible future lines of research are the improvement of the keys management system of the library. Our library uses RSA keys for attestation protocol that must be loaded in TPM. However the size for key storage in the TPM is very limited, then it must be carefully managed to avoid arisen space problems. The key management of our library might be improved, that is, our library handles the keys in such a way that only one key is loaded in TPM. Therefore, keys are loaded when will use and downloaded after they are used. This procedure is not very efficient due to the many key transactions done. We propose the use a mechanism that allows to download the same key that we will use in next step, but this is an open field for future researching. A different approach in the key management lies on caching these keys. Thus, several keys can be loaded simultaneously in the TPM making the management system more flexible and efficient. However, this approach presents some lacks. For instance, some kind of key replace policy might be established to determine which key is removed for a new one cache. Nevertheless, this task is out of the scope of this paper and we only propose as future researches. Another future line is to extend the library with new functionalities to secure migration services to provide of concurrency. That is, the secure migration service implemented in the library provides secure migration to a remote container, but they handle a unique request at the same time. Therefore, when the migration request arrives while migration is actually performed those are refused. This fact happens due to the TPM key management mentioned above. A possible extension of the library is to provide of a secure migration service with the capability to handle simultaneous requests. Finally, we propose the implementation of the SecMiLiA in a different tamper resistant hardware such as the smartcard to provide a proof of concept of the versatility of this approach.

Acknowledgments

Work partially supported by E.U. through projects SERENITY (IST-027587) and OKKAM (IST- 215032) and DESEOS project funded by the Regional Government of Andalusia.

References

1. Trusted computing group: Tcg specifications (2005), `https://www.trustedcomputinggroup.org/specs/`
2. Carl, A.G., Homeier, P., Nettles, S.: Infrastructure for proof-referencing code. In: Proceedings Workshop on Foundations of Secure Mobile Code (1997)
3. Collberg, C.S., Thomborson, C.: Watermarking, tamper-proofing, and obfuscation - tools for software protection. University of Auckland Technical Report 170 (2000)
4. Harrison, C., Chess, D., Kershenbaum, A.: Mobile agents: Are they a good idea?. IBM Research Report (1995)
5. Karnik, N.: Security in Mobile Agents systems. PhD thesis, Department of Computer Science, University of Minnesota (1998)
6. Maña, A.: Protección de Software Basada en Tarjetas Inteligentes. PhD thesis, University of Malaga (2003)
7. Maña, A., Muñoz, A., Serrano, D.: Towards secure agent computing for ubiquitous computing and ambient intelligence. In: Indulska, J., Ma, J., Yang, L.T., Ungerer, T., Cao, J. (eds.) UIC 2007. LNCS, vol. 4611, pp. 1201–1212. Springer, Heidelberg (2007)
8. Necula, G.: Proof-carrying code. In: Proceedings of 24th Annual Symposium on Principles of Programming Languages (1997)
9. Wahbe, R., Lucco, S., Anderson, T.E., Graham, S.L.: Efficient software-based fault isolation. In: Proceedings of the 14th ACM Symposium on Operating Systems Principles, pp. 203–216 (1993)
10. Roth, V.: Mutual protection of cooperating agents. In: Vitek, J. (ed.) Secure Internet Programming. LNCS, vol. 1603. Springer, Heidelberg (1999)
11. Stern, J., Hachez, G., Koeune, F., Quisquater, J.J.: Robust object watermarking: Application to code. In: Banerjee, U., Gelernter, D., Nicolau, A., Padua, D.A. (eds.) LCPC 1993. LNCS, vol. 768, pp. 368–378. Springer, Heidelberg (1994)
12. Yee, S.B.: A sanctuary for mobile agents. In: Secure Internet Programming (1999)
13. Bozga, L., Lakhnech, Y., Perin, M.: Hermes, a tool verifying secrecy properties of unbounded security protocols. In: Hunt Jr., W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 219–222. Springer, Heidelberg (2003)
14. Foundation for Intelligent Physical Agents: FIPA Abstract Architecture Specification (2002), `http://www.fipa.org/specs/fipa00001`
15. Muñoz, A., Maña, A., Harjani, R., Montenegro, M.: Agent Protection based on the use of cryptographic hardware. In: Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference, COMPSAC 2009 to be held in Seattle,Washington, July 20-July 24, IEEE, Los Alamitos (2009)

Security and Scalability of Remote Entrusting Protection

Vasily Desnitsky and Igor Kotenko

St. Petersburg Institute for Informatics and Automation (SPIIRAS)
39, 14 Linija, St. Petersburg, Russia
{desnitsky,ivkote}@comsec.spb.ru

Abstract. The paper outlines to the problem of correlation between security and scalability of software protection against tampering based on the remote entrusting principles. The goal of the paper is to propose a technique allowing choosing the most effective combination of different protection methods to apply. The technique is aimed at finding a trade-off between performance of the protection mechanism and its security, ensuring both a necessary security level and an appropriate scalability. The technique encompasses the evaluation of particular protection methods belonging to the whole protection mechanism and getting quantitative metrics of their performance and security level.

Keywords: Remote entrusting, performance analysis, security analysis, combination of protection methods.

1 Introduction

One of the most important requirements to software protection mechanisms is to provide a proper performance and scalability besides its security (attack resistance). This requirement is really necessary to implement to be able to use the mechanism in practice. Currently, software protection means, based on the client-server architecture, assuming that the server has to ensure correct service for a great number of clients working simultaneously, have not spread widely because of scalability problem.

The mechanism of software protection based on the remote entrusting, proposed in the RE-TRUST Project [9], is aimed at discovering the unauthorized modifications of a client program functioning in potentially hostile environment. This mechanism assumes a client program, to be protected, is executed within untrusted client environment, and a trusted entity is located on a safe host. According to the remote entrusting scenario [5], the protection mechanism uses different software (SW) and hardware (HW) based protection methods (Tamper Resistance methods, TR methods) as well as their combinations. Each of them being embedded into the whole protection mechanism represents some specific type of defense of a target application against tampering. The majority of TR methods assumes those implementation is shared between the client and the trusted server side. For instance, as one of TR methods, check sums are computed on clients and then delivered and checked on the server.

According to the remote entrusting principles, the important aim of the protection is to minimize the server computations to make the mechanism more scalable. Otherwise, if for every client the trusted server has to fulfill a lot of resource consuming

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 298–306, 2010.

computations, the support of a great number of clients could appear to be problematic and practically infeasible. Thus, the problems of performance and scalability arise.

This work is positioned at the conjunction of two research directions – security analysis and performance analysis of software protection methods. In contrast, the existing works address and estimate, for the most part, the security and performance of cryptographic protection methods, which better do for formal evaluation techniques [8, 13], or these properties of particular security protocols or tools [4, 11].

The paper aims for reaching the trade-off between security and scalability within the problem of SW protection based on remote entrusting principles. The remote entrusting mechanism is based on client-server architecture and uses a bundle of protection methods, which essentially differ from each other by protection principles and are characterized by diverse security and resource consuming requirements. Therefore, the estimation of such protection methods turns out to be a problem of specific character, which should have an acceptable solution. For instance, a great heterogeneity and disparateness of Software Guards [1] and Barrier Slicing [6] protection methods stipulate difficulty of producing a unified approach to evaluate their security strength and performance. Thus, in contrast to more conventional investigations (i.e. analysis of security level and performance penalties of various crypto ciphers, hash functions, etc.), the evaluation of TR methods in question and their combinations appears to be a weakly investigated task.

The paper is structured as follows. *Section 2* considers shortly TR methods and remote entrusting principles. *Section 3* outlines the proposed problem definition and analysis. In *section 3*, we formalize the task to be solved. *Section 4* contains the empirical studies focused on evaluating the performance and security level. *Conclusion* surveys the paper results and future research directions.

2 Tamper Resistance Methods and Remote Entrusting Principles

In the paper we differentiate two notions: a protection mechanism and a TR or protection method. By the protection mechanism we mean the protection mechanism against tampering based on remote entrusting principles, which includes a combination of different TR methods. In Table 1 some TR methods being applied within the protection mechanism are referenced to. The complete list and description of the protection methods are considered in [9].

Table 1. Examples of TR methods used in the protection mechanism

TR method	References
Control Flow Checking	[14]
Invariant Checking	[10]
Obfuscation techniques	[7]
Checksum monitoring	[2]
Crypto Guards	[1]
Barrier Slicing	[6]
Orthogonal and Continuous Replacement	[5]

Each TR method used within the mechanism implements one or several remote entrusting principles from the following list:

– *Remote attestation*. The principle assumes embedding a specific software component (monitor) into the client program. The monitor gathers data characterizing the program dynamic state and sends them to the trusted server for their checking. The principle is realized by means of such methods as invariant checking, control flow checking, checksum monitoring and others [1, 2, 7, 10, 14]. This protection principle supposes the fulfillment of a special detection function on the server.

– *Code splitting*. This principle lies in the fact that some code segments of the client program are extracted and transferred to the server. The goal here is to find and protect in such a way the most crucial parts of code. As a result an attacker can not directly access the processes running these code segments and, hence, is not able to influence them. As an example, barrier slicing method [6] implements this principle.

– *Dynamic replacement*. First, the principle comes to regular replacement of the monitor embedded into the client program. Second, it implements replacement of some program components critical from the security viewpoint. Periodic replacement in both considerations targeted at impediment to attacks on the protection mechanism and protected program. A representative example of this principle is orthogonal replacement method [5]. This method supposes realization of replacement with by means of creating mutually independent (orthogonal) versions of the software component on a basis of various obfuscation techniques.

In contrast to existing protection mechanisms such as Pioneer [15], SWATT [16], Genuinity [12] and some others, which accomplish software protection on the basis of client-server architecture and particularly implement remote attestation principle, the proposed protection method [9] is remarkable for a dynamic character of protection. This dynamism is described by the following properties: dynamic change of a bundle of applied TR methods; dynamic installation and enforcement different TR methods modules on the fly without suspending the protection process. The choice of particular TR methods is fulfilled reasoning from their characteristics of resource consumption and protection strength they provide.

3 Problem of Trade-Off between Security and Performance

The problem of achieving the reasonable trade-off between security and scalability is in the fact that, in addition to granting the proper security level, the protection mechanism has to be quite scalable to support protection for a sufficiently great amount of clients. If the mechanism is insufficiently scalable, its effectiveness will appear to be close to zero, since it can not be exploited in practice. The actions, which influence the mechanism's scalability, are primarily those ones that are fulfilled on the trusted server, i.e. verification functions and other procedures supporting the TR methods. The complexity of these actions grows proportionally to the amount of clients being served. Therefore reaching a good scalability requires using first of all those TR methods that do not contain any complex, resource consuming computations within the trusted entity.

By the scalability aim we mean a requirement that the dependency between the computational complexity of the needed actions on the trusted server and the quantity of clients being fulfilled simultaneously should be close to a linear or even constant function. It is obvious that the constant dependency is not feasible in practice, however the closer the dependency to constant one, the better scalability the mechanism reaches. More concrete, the scalability of the whole protection mechanism comes to the issue of scalability of each particular TR method.

The approach presented in the paper consists of solving the following tasks:

- *Evaluation of resources* consumed by each TR method on the trusted server.
- *Evaluation of security (attack resistance) level* of each TR method.
- *Choosing the most effective (optimal) combination* of TR methods for implementation under specific restrictions on available server resources.

As a result, an optimal is a combination of TR methods that allows achieving the highest scalability of the mechanism, having the proper security level ensured.

4 Problem Statements

For convenience and uniformity, let us describe below the formal expressions specifying the problem statement to select TR methods.

(1) Let M be a set of all *TR methods being realized within the protection mechanism*:

$$M = \{ m_{1,} m_{2, \dots} m_n \},$$

The set $M = \{ m_i \}$ is defined just as an enumeration of all the used protection methods m_i, $i = 1,\dots,n$.

(2) *Performance* determined by resource consumption function can be defined as p: $M \rightarrow P$, where P is a subset of $\mathbf{R}^r$ – space of vectors, where r is a number of the server resource types. p matches each protection method to a vector of values of its resource specific metrics. For each resource type r a constraint C[r] characterizing size of this resource is determined as well. Thus, for each protection method m_i, its resource consumption could be represented as a vector

$$(p^1(m_i) , p^2(m_i) ,\dots, p^r(m_i)).$$

(3) *Security level* is defined as a function determining a degree of provided protection for each TR method: s: $M \rightarrow S$, where S – a subset of **R** characterizing the security of different TR methods $\{m_i\}$, $i = 1,\dots,n$, from the set of selected (used) methods.

(4) *The problem statement:*

The common goal, we would like to achieve, is represented as:

$$\begin{cases} \sum\limits_{i:m_i \in \tilde{M}} p(m_i) \rightarrow \min\limits_{\tilde{M} \in 2^M} , \\ \sum\limits_{i:m_i \in \tilde{M}} s(m_i) \rightarrow \max\limits_{\tilde{M} \in 2^M} . \end{cases}$$

According to this formula, it is required to find a combination $\tilde{M}$ of TR methods that allows minimizing the total rcsource consumption and at the same time maximizing the total security level. In general case, the goal, we would like to achieve, is a multi-criterion optimization problem, which we suggest to bring to a single criterion onc.

As resource consumption function is defined by us as a vector function, wc suppose here to minimize some norm of it. There are several possible *definitions of the norm* in the space of resource consumption vector functions:

- A single component that is thc most critical. Here, $\| \sum p(m_i) \|$ equals to thc value of minimal component $p(m_i)$.
- A distance betwecn the vector of the total resourcc consumption and the constant vector $C \in R^r$ characterizing server resources available for protection methods.

Let us consider a refined statement of the problem we solve, which is expressed by the following formulas:

$$\begin{cases} p(\tilde{M}) \to \min\limits_{\tilde{M} \in 2^M} \\ \sum\limits_{m_i \in \tilde{M}} p^r(m_i) \leq C[r] \quad \forall r:\ r \in R \\ \sum\limits_{m_i \in \tilde{M}} s(m_i) \geq \hat{s} \end{cases} .$$

Here by $p(\tilde{M})$ wc mean the total valuc of resource consumption metric for a combination $\tilde{M}$ of TR methods, whereas $\hat{s}$ denotes a constraint determining the minimal due security level thc methods should provide. Thus, the task to be solved is to choose a sct of protection mcthods that the resource consumption function to be no more than a spccific constant, having the propcr total security provided. Constant $\hat{s}$ is assumed to be detcrmined by means of both empirical study and theoretical analysis of resistance of the protection mcthods. In practice, the precise value could be specified by thc designer/administrator of the system and supposed to change.

(5) *Computation of performance and security metrics:*

Consider how $p(m_i)$ and $s(m_i)$ values could be detcrmined for each protection method. Estimation of resourcc consumption is represented by the following approaches having both theoretical and empirical peculiarities:

- On the thcoretical lcvcl, each TR method is subjected to analysis and its model, representing its implcmentation, is constructed. Such a model contains merely those operations that are the most important for performance vicwpoint. Resource consumption metrics for protection mcthod assessment are dcveloped as well.
- On the empirical lcvcl, both the software realization of this model on a high-level programming language and the procedures to mcasure the resource

consumption metrics are completed. Computation of group metrics p is fulfilled by means of the following formulas:

$$p^r(\tilde{M}) = \sum_{i:m_i \in \tilde{M}} p^r(m_i)$$

$$p\ (\tilde{M}) = \min\{p^r(\tilde{M}) \mid r \in R\}.$$

For each m_i and r the values of $p^r(m_i)$ are obtained experimentally, whereas the values $p\ (m_i)$ and $p\ (\tilde{M})$ are calculated analytically.

The group metrics are calculated by means of the values of the single metrics. Therefore the quantity of experiments, conducted for group metric computation, represents a linear function of the amount of protection methods, instead of an exponential one otherwise.

The difficulty of $s(m_i)$ determining is in the fact that security according to its nature is a qualitative characteristic of a protection method. Meanwhile, our task is supposed to contain also granting some quantitative character to security. The aim is to get a possibility in different cases to make a choice of the most preferable (from security viewpoint) comhinations of protection methods.

In general case protection level of a TR method m_i is meant as a complexity of accomplishment of an attack aimed at its compromise. This approach includes attack complexity estimation for each TR method. In practice, however it is considered to be infeasible due to the complexity of analysis involved, including the complexity of estimating the attacker's cognitive processes, which are the core elements in the process of attack fulfillment by the intruder.

We have proposed a technique hased on expert judgments, which is supposed to collect and process the opinions of experts with use of system analysis methods. Each expert states a number (from 1 to 10) to each protection method. An advantage of this approach is that in its work it takes into consideration all knowledge and experience accumulated hy all experts. In contrast to the previous approaches this one does not suppose any generalizations, which ultimately introduce extra inaccuracy into the outcome. Values $s(m_i)$ are obtained by means of questioning of experts and calculating averaged values for each protection method, taking into account both a priori and a posteriori competence of every expert in the field of a particular protection method. Values of group metrics of protection methods for combinations $s(\tilde{M})$ are calculated using the following formula:

$$s(\tilde{M}) = \sum\nolimits_{m_i \in \tilde{M}} s(m_i)\ .$$

5 Empirical Study

The technique of combining different protection methods consists of the following main stages: (1) performance evaluation, (2) security evaluation, and (3) determining the most effective combination of protection methods. As input data, a set of TR methods is used. As output data a set of combinations of TR methods is produced.

To evaluate the performance of TR methods, a SW prototype, implementing several of them, including control flow checking (m_1), invariant checking (m_2), barrier slicing (m_3) and orthogonal replacement (m_4) has been realized. On a base of this prototype some measurements of resource consumption have been conducted. For each TR method, a highest quantity of clients that can be served simultaneously is evaluated. The value of intensity of the server loading is also measured. This intensity is a ratio between the time the server's processor is loaded, when carrying out the protection method, and the entire time reserved for the method.

Fig. 1 demonstrates some results of the experiments, including the evaluation of these four protection methods. Fig. 1 shows dependencies between the consumption of the resource r and the amount u of clients which can be served. Here the resource $r = 1$ determines the metric of processor loading intensity (p^1), whereas C[1] represents a constraint of the whole available resource volume. $p^1(\tilde{M})$ denotes the metric of resource consumption for a combination $\tilde{M}$ of protection methods.

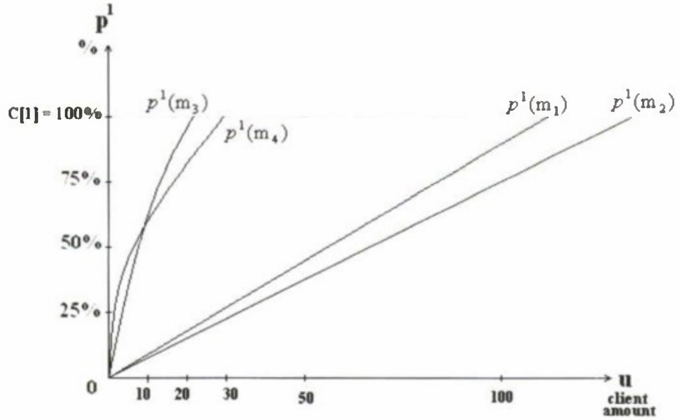


Fig. 1. Dependencies between value of resource consumption and amount of clients

The experiments have shown that in practice one should distinguish one-time procedures being accomplished by the server, when new client is connecting (in particular, actions on connection establishment, client authentication or crypto key agreement), and the regular actions on verification of clients. Hence, one should avoid simultaneous mass client connections to avoid strong peak loading.

In experiments during expert questioning [3], the data from ten security experts have been received.

Table 2 shows the generalized estimations of security level for some of the methods investigated. One should take into account a relativity of these results. Surely, this evaluation technique can not be exploited as a proof of adequacy of the protection mechanism and especially to compare the strength of this mechanism with any other SW protection means. The technique under consideration represents relatively rough solution to evaluate protection methods, which adequacy is sufficient for TR methods

Table 2. Results of security evaluation of protection methods

TR method	Security level
Barrier Slicing	9,0
Orthogonal Replacement	7,8
Continuous Replacement	7,1
Crypto Guards	6,2
Control Flow Checking	4,3
Invariant Checking	3,3
Obfuscation technique: opaque predicates	1,3

Table 3. Experimental values of metric processor loading intensity

M u	m_1	m_2	m_3	m_4
10	8,7	7,2	59,9	61,1
20	17,4	14,3	95,6	81,5
25	21,8	37,8	–	92,0
50	43,5	35,7	–	–
100	87,1	71,3	–	–

combination task being solved. Thus, as a whole, this solution can be regarded as a supplement to security evaluation techniques based on formal approaches, which have their own drawbacks, particularly, as a rule, they are characterized by a significant complexity in implementation and further analysis.

Thus, the technique, forming the search of optimal combinations of TR methods, comes to determining some numerical data characterizing, first, the performance for each method and, second, its security level.

Table 3 contains the experimental values of metric p^1 obtained for different amount of clients (u) and different TR methods - control flow checking (m_1), invariant checking (m_2), barrier slicing (m_3) and orthogonal replacement (m_4).

The optimization problem settled in Section 4 is tackled by an improved exhaustive search, supposing a restriction of combinations under consideration, cutting those ones that are deliberately not optimal. Note, for methods m_3 and m_4 for some u values the metric values are not specified, that is for this amount of the current size of r^1 this amount of clients can not be served.

6 Conclusion

In the paper we have proposed the technique determining how to combine various protection methods based on remote entrusting. The technique allows addressing the problem of reaching the compromise between scalability and security. On account of objective difficulties of correct security evaluation, we have chosen the technique of security evaluation based on expert judgments. The technique for evaluating the performance of protection methods comes to empirical study, where resources consumption values are obtained as metric values. Experiments to compute the values of performance metrics as well as to question the experts and process the received judgments on security strength of protection methods were carried out. As a future work, we supposed to search and construct more comprehensive and precise techniques of performance and security evaluation and perform more detailed experiments to choose efficient combinations of protection methods.

Acknowledgments

This research is partly funded by the EU under RE-TRUST and SecFutur projects, the grant of the Russian Foundation of Basic Research (Project No.10-01-00826) and Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (Contract No.3.2).

References

1. Atallah, M., Bryant, E., Stytz, M.: A survey of Anti-Tamper Technologies. The Journal of Defence Software Engineering (2004)
2. Barbara, D., Goel, R., Jajodia, S.: A checksum-based corruption detection technique. Journal of Computer Security 11(3) (2003)
3. Beshelev, S.D., Gurvich, A.G.: Mathematical-statistical methods for expert Judgments. Statistika (1980)
4. Chappell, B.L., Marlow, D.T., Irey, P.M., O'Donoghue, K.: An Approach for Measuring IP Security Performance in a Distributed Environment. In: Rolim, J.D.P. (ed.) IPPS-WS 1999 and SPDP-WS 1999. LNCS, vol. 1586. Springer, Heidelberg (1999)
5. Ceccato, M., Preda, M., Majumdar, A., Tonella, P.: Remote software protection by orthogonal client replacement. In: The 24th ACM Symposium on Applied Computing (2009)
6. Ceccato, M., Preda, M., Nagra, J., Collberg, C., Tonella, P.: Barrier Slicing for Remote Software Trusting. In: The IEEE International Working Conference on Source Code Analysis and Manipulation, Paris, France (2007)
7. Collberg, C., Thomborson, C.: Watermarking, tamper-proofing, and obfuscation tools for software protection. IEEE Transactions on Software Engineering 28 (2002)
8. Freeman, W., Miller, E.: An Experimental Analysis of Cryptographic Overhead in Performance-Critical Systems. In: The 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (1999)
9. FP6 Project RE-TRUST, `http://www.re-trust.org`
10. Godoy, G., Tiwari, A.: Invariant Checking for Programs with Procedure Calls. In: Palsberg, J., Su, Z. (eds.) SAS 2009. LNCS, vol. 5673, pp. 326–342. Springer, Heidelberg (2009)
11. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley-Interscience, New York (1991)
12. Kennell, R., Jamieson, L.H.: Establishing the genuinity of remote computer systems. In: The 12th USENIX Security Symposium, Washington, DC, USA (2003)
13. Menasce, D.A.: Security performance. IEEE Internet Computing 7(3) (2003)
14. Oh, N., Shirvani, P.P., McCluskey, E.J.: Control-flow checking by software signatures. IEEE Transactions on Reliability 51 (2002)
15. Seshadri, A., Luk, M., Shi, E., Perrig, A., Doorn, L.V., Khosla, P.: Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In: The Twentieth ACM Symposium on Operating Systems Principles. ACM Press, New York (2005)
16. Seshadri, A., Perrig, A., Doorn, L.V., Khosla, P.: SWATT: SoftWare-based ATTestation for Embedded Devices. The IEEE Symposium on Security and Privacy (2004)

A Novel Genetic Approach to Provide Differentiated Levels of Service Resilience in IP-MPLS/WDM Networks[*]

Wojciech Molisz and Jacek Rak

Gdansk University of Technology, G. Narutowicza 11/12,
PL-80-233 Gdansk, Poland
{womol,jrak}@eti.pg.gda.pl

Abstract. This paper introduces a novel class-based method of survivable routing for connection-oriented IP-MPLS/WDM networks, called MLS-GEN-H. The algorithm is designed to provide differentiated levels of service survivability in order to respond to varying requirements of end-users. It divides the complex problem of survivable routing in IP-MPLS/WDM networks into two subproblems, one for each network layer, which enables finding the solutions in a relatively short time. A genetic approach is applied to improve the quality of results by solving the problem iteratively.

Modeling results show that, after a reasonable number of iterations, a good solution (up to 22.55% better than the initial one) is found and further improvement is hardly possible.

Keywords: service survivability, IP-MPLS/WDM networks, routing, differentiated levels of service resilience, genetic algorithms.

1 Introduction

Backbone networks are migrating from synchronous transmission infrastructure to next generation, high-traffic-volume data (e.g.: IP-MPLS or IP/Ethernet) over optical transport networks (OTNs). By applying wavelength division multiplexing (WDM), OTNs are capable of carrying many independent channels (currently 160 or 320), over a single optical fiber with the fastest channels supporting a data rate of 40 Gbps. Fiber cuts (the most typical network outages) may lead to service disruption and huge data and revenue losses. *Survivability*, i.e. capability to deliver essential services in the face of failure, or attack, is a key concern in network design. There are two approaches for providing survivability of connection-oriented IP-over-OTNs: *protection* and *restoration* [15]. In the protection approach, *working lightpaths* (being sequences of wavelengths over an optical network with fully optical processing at intermediate nodes) are protected by the pre-computed *backup paths*, applied in the case of working path failures. Restoration finds dynamically a new path, once a failure has

[*] This work was partially supported by the Ministry of Science and Higher Education, Poland, under the grant PBZ-MNiSW-02-II/2007.

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 307–320, 2010.

occurred. Usually we distinguish either path protection/restoration, or link protection/restoration against a single link or node failure. However, intermediate solutions also exist, like e.g. *area protection* [9], *partial path protection* [17], or *segmented shared protection* [16].

1.1 Related Works

Majority of publications focus on providing survivability in one (usually optical) layer. Recently, several papers have appeared on survivability models of IP-over-WDM networks. Sahasrabuddhe, Ramamurthy and Mukherjee [15], assuming that each backup path in the WDM layer is arc-disjoint with the respective primary path, analyzed protection in the WDM layer, and restoration in the IP layer. They proposed four integer linear programming (ILP) models and heuristics to find solutions. Pickavet et al. [13] discussed three approaches to interworking between the network layers and two efficient coordinated multilayer recovery techniques.

Ratnam, Zhou and Gurusamy [14] addressed the problem of efficient multilayer operational strategies for survivable IP-over-WDM networks and proposed several joint multiple layer restoration schemes with intra-layer and inter-layer signaling and backup resource sharing. Bigos et al. [4], and Liu, Tipper and Vajanapoom [8] described various methods for spare capacity allocation (SCA) to reroute disrupted traffic in MPLS-over-OTN. Cui et al. [5] proposed a multilayer restoration and routing in IP-over-WDM networks (called EROTRIP) with the bottom-up scheme and GMPLS token for signaling between the layers. Recently, Harle and Albarrak [7] proposed a model of differentiated survivability in a GMPLS-based IP-over-OTN network with cooperation mechanisms between control planes in different layers.

1.2 Outline

We consider here a survivable IP-MPLS-over-OTN-WDM network protected against a single node failure. Demands for IP flows are given. We assume M service classes, numbered from 0 to M-1. Class m = 0 represents the demands for which all service recovery actions must be performed as fast as possible (i.e. in the WDM layer). For other service classes, the values of IP-MPLS restoration time may increase. In the first stage of our algorithm, for each service class we find a node-disjoint pair of working and backup label switched paths (LSPs). Then we group the IP demands into service classes on IP links. In the next stage we map working LSPs of each class onto protected lightpaths according to available capacities of optical links. The scope of WDM protection depends on the service class number: in the highest class m = 0, each two adjacent WDM links of the working lightpath are protected by a backup lightpath, while in the lowest class (m = M-1) with no backup lightpaths, all the recovery actions must be performed at the IP-MPLS layer. We assume a bottom-up restoration strategy. If a working path consists of more than one link, then a failure of the lightpath transit node can be restored in the optical layer. Connections which cannot be restored in the WDM layer, should be restored in the IP layer. All the optimization models are NP-complete, since their simpler version – the task to find $|D|$ working paths in capacitated networks in a single network layer is NP-complete [11]. Therefore we propose the novel genetic approach, extending the one of [10].

The rest of the paper is organized as follows. The survivable routing problem is sketched in Section 2. Heuristic algorithms are then developed to solve the problems: the MLS-H algorithm to find initial solutions, and MLS-GEN-H to improve them. Modeling assumptions are described in Section 3. Results discussed in Section 4 show that it is possible to decrease the total network cost by utilizing the technique of genetic algorithms.

2 Survivable Routing of IP-MPLS Demands in the IP-MPLS/WDM Network

Due to the complexity of the original problem of integrated survivable routing, similar to our work [10], we divide here the problem of survivable IP-MPLS/WDM routing into two following subproblems:

a) survivable IP-MPLS routing consisting of determining the IP-MPLS virtual topology and finding the survivable routing of IP-MPLS demands,
b) survivable WDM routing (lightpath routing and wavelength assignment).

Our goal is to provide the differentiated levels of service resilience in order to respond to varying requirements of end-users. This differentiation is defined in terms of the values of service recovery time and the frequency of performing the time-consuming recovery actions in the IP-MPLS layer. That's why we introduce M service classes, numbered from 0 to M-1. Class $m = 0$ comprises demands, for which the time of service recovery and the frequency of recovery actions in the IP-MPLS layer must be minimized. For other service classes, these values are allowed to increase.

In order to achieve our goal, the number of working LSP links should depend on the service class m, and is determined as:

$$\hat{\delta}_m = \left\lceil \frac{|\mu_t| - 1}{M - 1} \times m + 1 \right\rceil \tag{1}$$

where: $|\mu_t|$ is the number of arcs of the end-to-end shortest path between the source p_t and destination q_t nodes of the t-th IP-MPLS demand
m is the class of a demand and M is the number of service classes.

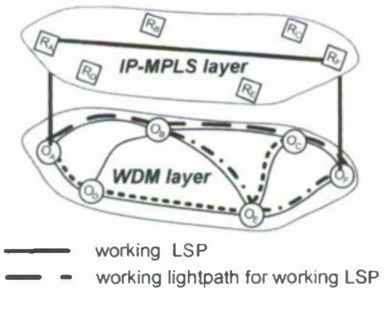


Fig. 1. Example IP-MPLS/WDM survivable routing (class $m = 0$)

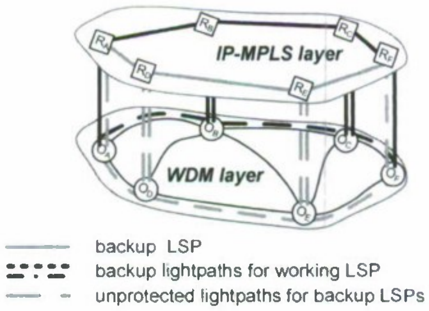

R_A
R_B
R_C
R_D
R_E
R_F
IP-MPLS layer
O_A
O_B
O_C
O_D
O_E
O_F
WDM layer
backup LSP
backup lightpaths for working LSP
unprotected lightpaths for backup LSPs

Fig. 2. Example IP-MPLS/WDM survivable routing (class $m = M$-1)

It is clear from the formula (1), that any working LSP for the class $m = 0$ demand is established by a direct IP-MPLS link, as shown in Fig. 1. This implies in turn that no time-consuming IP-MPLS recovery actions will take place. On the contrary, for the class $m = M$-1, each link of the working LSP will be mapped onto a single-link WDM lightpath (Fig. 2), implying frequent recovery actions in the IP-MPLS layer.

Similar to [10], the objective of the subproblem (a) is to find the survivable LSPs, where each working LSP is protected by the end-to-end backup LSP having no common transit nodes with the respective working LSP.

Fast service recovery in the WDM layer is achieved here by limiting the scope of lightpath protection. For that purpose we determine the number of backup lightpaths protecting the given working lightpath as:

$$\delta_m = \left\lceil -\frac{|\pi_r|-1}{M-1} \times m + |\pi_r| - 1 \right\rceil \tag{2}$$

where: $|\pi_r|$ is the number of arcs of the shortest path between the lightpath end-nodes; all the other symbols have the same meaning as in (1).

From the formula (2), one can observe that δ_m decreases linearly with the increase of service class number m. In particular, it means that, for the class $m = 0$, any backup lightpath protects two adjacent links of the working lightpath, as shown in Fig. 1, while, for the class $m = M$-1, there is no backup lightpath for a given working lightpath and all the recovery actions must be performed in the IP-MPLS layer (Fig. 2). However, due to limitations on the number of working LSP links in the IP-MPLS layer (implying the decrease of the length of the working lightpath with the increase of the service class number m, as given in Eq. 1), the real scopes of WDM protection (measured in kilometers of fibers) remain at the same low level, independent of the service class number. This in turn provides fast service recovery in the WDM layer independent of the service class number. The only exception is for the class $m = M$-1 with no backup lightpaths in the WDM layer.

The backup LSPs are grouped into service classes and mapped onto the unprotected lightpaths. The basic MLS-H algorithm is given in Fig. 3.

Steps 1÷3 arc responsible for determining the survivable IP-MPLS routing, while Steps 4÷6 are used to find the survivable routing in the WDM layer. Since each subproblem considered here is NP-complete, we have used only the heuristic approach in computations. Due to the limitations on the size of the paper, the respective ILP formulations may be found in the electronic version at [20].

However, finding the solution to the survivable IP-MPLS routing problem in each network layer separately, as given in the MLS-H algorithm from Fig. 3, certainly leads to suboptimal solutions. To overcome this problem, in this paper we propose to use the metaheuristic approach based on genetic programming to improve the quality of results by solving the problem iteratively. Any *genetic algorithm* is a domain-independent approach based on the mechanisms of natural selection and natural genetics [6]. Its main advantage is the ability to perform the parallel search when finding the best solution as well as the adaptability to the problem.

```
INPUT   WDM layer topology $\Gamma = (N, A)$, where $N$ and $A$ are the sets of nodes and arcs;
        A set $|D_{IP}|$ of IP-MPLS demands, each demand $d_t$ given by a quadruple
        $d_t = (p_t, q_t, m, f(\mu_t))$, where $p_t$, $q_t$, $m$, $f(\mu_t)$ are: the source node, the destination
        node, the service class number and the requested capacity, respectively.
OUTPUT  Survivable multilayer routing of demands

Step 1  Create the matrix $\Xi$ of costs $\xi_h$, each cost $\xi_h$ equal to the length of WDM arc $a_h$.
Step 2  For each demand $d_t$, find a pair of working and backup LSPs using Bhandari's
        algorithm [3] and the matrix $\Xi$ of arc costs.
Step 3  Divide the working LSPs into $\delta_m$ regions (Eq. 1) and replace each part of the
        working LSP, determined by the given region, with a direct IP-MPLS link.
Step 4  Find the working lightpaths carrying the IP-MPLS working paths, using the
        Dijkstra's algorithm [3] and the standard distance metrics, treating the aggregated
        flows from the same service class $m$ between the end-nodes of the IP-MPLS
        virtual links $v_r$, found in Step 3, as the demands for the WDM layer.
Step 5  Divide each working lightpath into $\delta_m$ regions, as given in Eq. 2, and provide
        each region with a dedicated backup lightpath. For that purpose, replace each
        part of the working lightpath, determined by the given region, with a pair of
        node-disjoint paths, found using Bhandari's algorithm*.
Step 6  Provide each aggregated IP-MPLS backup flow between the end-nodes $s_r$ and $t_r$
        of the IP-MPLS layer virtual link $v_r$ with the unprotected WDM lightpath*.

* if finding any lightpath is not feasible due to the lack of resources, then reject all the end-user
  demands, for which the respective paths were to be groomed into the given lightpath
```

Fig. 3. The basic MLS-H algorithm to find the survivable routing of IP-MPLS demands

Any genetic algorithm starts with finding *an initial population* of $|CH|$ *chromosomes*, each chromosome typically represented by a binary vector. Each next iteration is to find a new population of $|CH|$ chromosomes, by choosing the best ones from the current population as well as from the sets of $|CRS|$ and $|MUT|$ new chromosomes, obtained in the crossover and mutation operations, respectively. In a single *crossover* operation, two chromosomes, randomly chosen from the current population, are used to produce a new pair of chromosomes. Each time, a crossover point is selected randomly within the length of a chromosome, and the respective *genes* are exchanged with each other.

Another operation - *mutation*, unlike crossover, makes changes within an individual chromosome, randomly chosen from the set of $|CH|$ population chromosomes, rather than across a pair of chromosomes.

In order to adapt the genetic approach to solve the IP-MPLS/WDM survivable routing problem, the following assumptions were made:

Chromosome

A single chromosome was formed by a matrix Ξ of costs ξ_h of arcs $a_h = (i, j)$ used when finding the working and backup LSPs. The quality of a chromosome was measured in terms of the total link capacity utilization ratio by finding the solution to the respective IP-MPLS/WDM survivable routing problem, using the MLS-H algorithm (Fig. 3) with the costs ξ_h of network arcs a_h stored in the chromosome.

Initial Population

The initial population of $|CH|$ chromosomes was formed by $|CH|$ matrices Ξ of arc costs ξ_h, each matrix obtained by introducing the random modifications to the matrix Ξ of the reference costs ξ_h of WDM arcs a_h.

Crossover

This operation was performed on a pair of randomly chosen chromosomes ch_A and ch_B to obtain a new pair of chromosomes. A point of crossover was randomly determined first. The crossover operation assumed the exchange of parts of the symmetric matrix Ξ of arc costs, as given in Fig. 4.

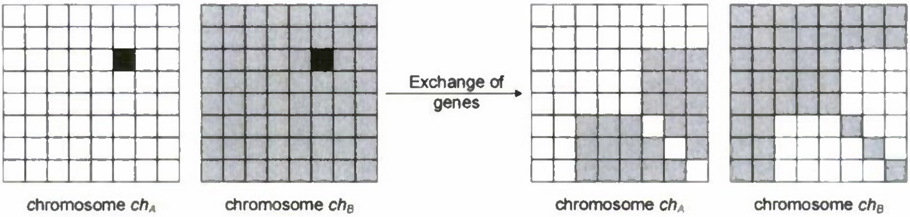


Fig. 4. Example crossover operation in MLS-GEN-H algorithm

```
INPUT     WDM-layer topology $\Gamma = (N, A)$; A set of IP-MPLS layer demands $D_{IP}$, each
          demand given by a quadruple $d_t = (p_t, q_t, m, f(\mu_t))$, where $p_t, q_t, m, f(\mu_t)$ are:
          the source node, the destination node, the service class number
          and the requested capacity, respectively; the number of iterations $ic$.
OUTPUT    Survivable multilayer routing of demands.

Step 1    Set $i = 0$.
Step 2    Create the initial population of $|CH|$ chromosomes.
Step 3    Obtain $|CRS|$ new chromosomes by applying the crossover operation, each
          such operation for a randomly chosen pair of existing chromosomes
          from the population.
Step 4    Obtain $|MUT|$ new chromosomes by using the mutation operation, each such
          operation for one chromosome randomly chosen from the population.
Step 5    Measure the quality of $|CRS|$ + $|MUT|$ new chromosomes by executing
          the MLS-H algorithm once for each given chromosome (matrix $\Xi$).
Step 6    Choose $|CH|$ out of $|CH|$ + $|CRS|$ + $|MUT|$ best chromosomes to form
          the new population.
Step 7    If $i = ic$ then return the best solution from the current population
          else
              Step 7.1   Set $i = i + 1$.
              Step 7.2   Go to Step 3.
```

Fig. 5. The MLS-GEN-H algorithm

Mutation

During a single mutation operation, a randomly chosen gene $ge_l = (i, j)$ of a given randomly chosen chromosome, being the cost of respective arc $a_h = (i, j)$, was assigned a random value from the range (0, ξ^{MAX}), where ξ^{MAX} was the length of the longest arc in the WDM layer. Since each cost matrix Ξ was symmetrical, the same random value was sct to the gene $ge_2 = (j, i)$.

The MLS-GEN-H algorithm is presented in Fig. 5.

The algorithm first creates the set of $|CRS|$ initial chromosomes (Step 2). Each iteration of the MLS-GEN-H algorithm is formed by thc execution of Steps 3÷7. In Steps 3÷4, $|CRS|$ and $|MUT|$ new chromosomes are produced during the crossover and mutation operations, respectively. In Step 5, the quality of each new chromosome is verified by executing the MLS-H algorithm from Fig. 3. Finally, the best $|CH|$ out of $|CH| + |CRS| + |MUT|$ chromosomes are chosen to form the new population. The algorithm terminates after reaching the given number of iterations defined by thc *ic* variable and returns the best solution from the last population.

The MLS-GEN-H algorithm has the polynomial computational complexity of $O(|N|^2)$, since in Step 2 it executes $ic\cdot(|CRS|+|MUT|)$ times the MLS-H algorithm of complexity $O(|N|^2)$ to check the quality of new chromosomes. Additionally, each operation of crossover and mutation requires $O(|N|^2)$ and $\Theta(1)$ time, accordingly.

3 Modeling Assumptions

The modeling was to evaluate the properties of the proposed approach regarding the following characteristics:

a) the average lcngth and thc avcrage number of links of working and backup paths,
b) the total number of brokcn connections and the average value of service rcstoration time, measured in a single simulation scenario.

The results additionally include the ratio of improvement in solution quality for the genetic approach as a function of the iteration number, measured as the decrease in the total number of channels, needed to provide the class-based survivable routing for the best solution in each next population. They are presented for four nctworks, namely, the European COST 239 Network, the Italian Network, thc NSF Network and the U.S. Long-Distance Network (see Figs. 6÷9).

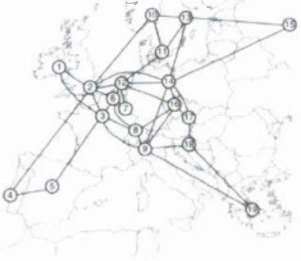


Fig. 6. European COST 239 Network [18]

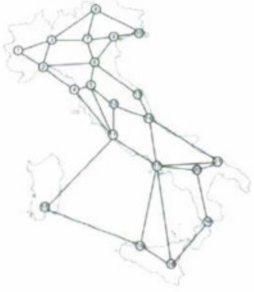


Fig. 7. Italian Network [1]

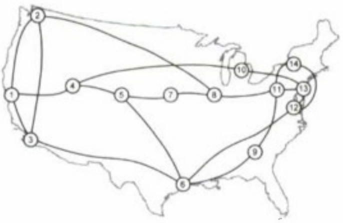


Fig. 8. NSF Nelwork [12]

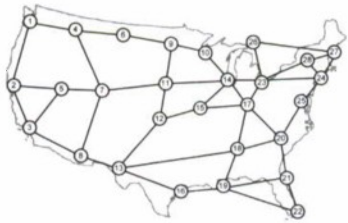


Fig. 9. U.S. Long-Distance Network [19]

All the WDM layer links were assumed to have 32 channels. Channel capacity unit was considered to be the same for all the links. Nodes had the intcgrated functionality of optical cross connects (OXCs) in the optical layer and of the IP-MPLS routers. OXCs were assumed to have a full wavelength conversion capability.

Time of service restoration in the WDM layer comprised: timc to detect a failure, link propagation delay, time to configure backup lightpath transit nodes and message processing delay at network nodes (including queuing delay). The recovery actions, that had to be finalized in the IP-MPLS layer, additionally included thc following:

- time to determine that the WDM layer is not able to restore the affected flow (i.e. the time of unsuccessful recovery in the WDM layer equal to the time needed to send the `NODE FAIL` message to the lightpath end-nodes)
- time to reroute the affected flow in the IP-MPLS layer comprising:
  - time to deteet the failure in the IP-MPLS layer τ_{TDF}^{IP}, whieh ineludes the time to transfer the reeovery token to the WDM layer. In simulations, the value of τ_{TDF}^{IP} = 20 ms was used, as defined in [2],
  - time to send the notification to the working LSP source node along the working LSP links about the failure, based on the aggregate transmission delay of the corresponding working lightpaths and message processing delay τ_{MPD}^{IP} = 20 μs at thc working LSP transit nodes (as given in [2]),
  - time to eonfigure, test and set up the forwarding table at the respeetive LSRs set to τ_{CNF}^{IP} = 10 ms (following [2]),
- time to aetivate the baekup LSP being the aggregate time to aetivate all the respective unprotccted lightpaths carrying the baekup LSPs.

For eaeh IP-MPLS layer connection, the following properties were assumed:

- demands from $M = 5$ service classes with protection against a single node failure,
- the demanded capacity equal to 1/8 of the WDM link channel capacity,
- protection against a single node failure,
- provisioning 100% of the requested bandwidth after a failure,
- a demand to assure unsplittable flows in both the IP-MPLS and the WDM layer,
- the distanee metrics and the Bhandari's algorithm [3] of finding k-node disjoint paths (here $k = 2$) in all path eomputations, except for the unproteeted lightpaths for the baekup LSP links, found by the Dijkstra's shortest path algorithm [3],
- the number of generated populations set to $ic = 1000$,

- the size each population equal to $|CH| = 20$ chromosomes,
- number of chromosomes achieved during the crossover and mutation operations in each iteration: $|CRS| = 10$ and $|MUT| = 10$, accordingly,
- percentage of chromosome genes changed during the mutation operation: 10%,
- type of mutation: random value insertion,
- the three-way handshake protocol of service restoration in the IP-MPLS and WDM layer (the exchange of `NODE FAIL`, `SETUP` and `CONFIRM` messages).

However, for the analyzed MLS-GEN-H algorithm, the time needed to perform $ic = 1000$ iterations using a Pentium IV 2.4 GHz workstation with 512 MB RAM was up to one week for a single demand set (for the case of the U.S. Long Distance Network with the size of a demand set equal to 100% node pairs). For this reason, for any of the investigated network, computations were done for a single demand set only.

The algorithm of a single modeling scenario is shown in Fig. 10.

Execute the following steps:

Step 1 Randomly choose $|D_{IP}|$ pairs (p_t, q_t) of nodes ($|D_{IP}|/M$ demands for each service class)*.

Step 2 Try to establish the survivable connections using the MLS-GEN-H algorithm.

Step 3 Store the ratio of link capacity utilization and the lengths of paths.

Step 4 u* times simulate random failures of single nodes. For each failure, restore the broken connections and memorize the values of connection restoration time.

* *in each scenario, $u = 100$ was assumed. The number of demands $|D_{IP}|$ was set to 25, 50, 75 or 100% of all the network node pairs chosen randomly, accordingly.*

Fig. 10. Research plan

4 Modeling Results

4.1 Average Path Lengths and Numbers of Links of Connection Paths

Fig. 11 shows the average lengths of working and backup LSPs as a function of the service class number, while Fig. 12 gives the respeetive numbers of LSP links. Table 1 shows the lengths of the 95% confidence intervals of the average path length. Independent of the serviee elass number m, the lengths of the IP-MPLS layer working and baekup paths remain at the same level, eharaeteristie to path proteetion scheme. The average number of IP-MPLS layer working path links, defined in Eq. 1, decreases with the decrease of the service class number m (Fig. 12), resulting in less frequent time-consuming recovery actions in the IP-MPLS layer for more important service classes. Since each link of the backup LSP is established as the one-hop lightpath, the average number of backup LSP links remains at the same level, independent of the service elass number.

Fig. 13 shows the average lengths of WDM layer lightpaths as a function of a service class number, while Fig. 14 gives the numbers of WDM layer path links. Table 2 presents the lengths of the 95% eonfidenee intervals of the average lightpath length. The average length of backup lightpaths remains at the same level, independent of the service class number. However, the average length of working lightpaths decreases

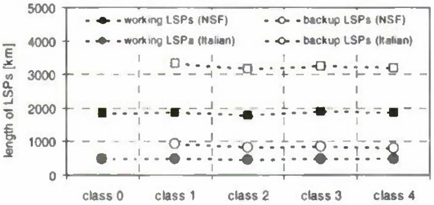


Fig. 11. Average length of LSPs

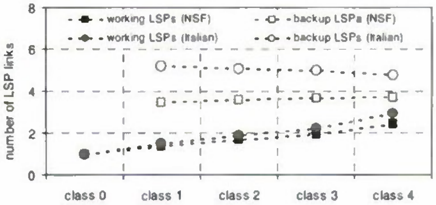


Fig. 12. Average number of LSP links

Table 1. Lengths of 95% confidence intervals for the average length of LSPs [km]

network	working LSPs					backup LSPs				
	class 0	class 1	class 2	class 3	class 4	class 0	class 1	class 2	class 3	class 4
NSF	160.30	161.38	173.98	176.46	175.28	-	204.80	179.38	181.08	175.13
Italian	32.34	31.84	33.08	33.35	33.37	-	78.08	51.60	59.74	50.26

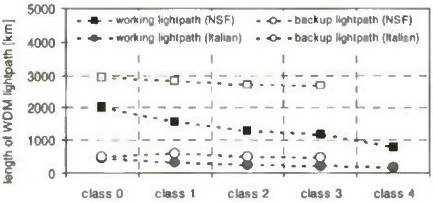


Fig. 13. Average length of lightpath

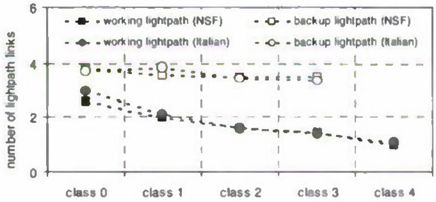


Fig. 14. Average number of lightpath links

Table 2. Lengths of 95% confidence intervals for the average length of lightpaths [km]

network	working lightpaths					backup lightpaths				
	class 0	class 1	class 2	class 3	class 4	class 0	class 1	class 2	class 3	class 4
NSF	185.10	151.11	128.69	121.62	56.28	170.67	231.90	212.76	322.21	-
Italian	27.15	18.89	12.32	11.15	6.61	15.75	32.78	24.61	26.79	-

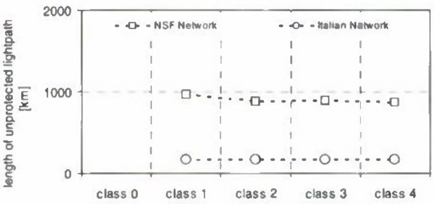


Fig. 15. Average length of unprotected lightpaths

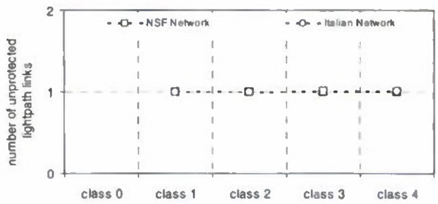


Fig. 16. Average number of unprotected lightpath links

with the increase of service class number. This is due to the fact that with the increase of the service class number, the number of IP-MPLS layer working LSP links increases, and each link of the working LSP is realized by a shorter WDM lightpath.

Fig. 15 shows the average lengths of unprotected lightpaths realizing the backup LSPs as a function of the service class number, while Fig. 16 gives the respective numbers of LSP links. Independent of the analyzed network, the average length of the unprotected lightpath, each unprotected lightpath realized by a direct WDM layer link (Fig. 16), remains at the same level for all the service classes.

4.2 Service Recovery Actions

Fig. 17 shows the aggregate numbers of service recovery actions for both the IP-MPLS and the WDM layer, measured in a single scenario. It shows that, with the increase of the service class number, the number of recovery actions in the IP-MPLS layer increases, while the number of recovery actions in the WDM layer decreases. For the class m = 0, it implies that the WDM-layer recovery actions are sufficient to handle all the failure cases and, as a result, they provide fast service recovery. For the other service classes, with the increase of the number of IP-MPLS working path transit nodes, the frequency of IP-MPLS recovery actions gets increased.

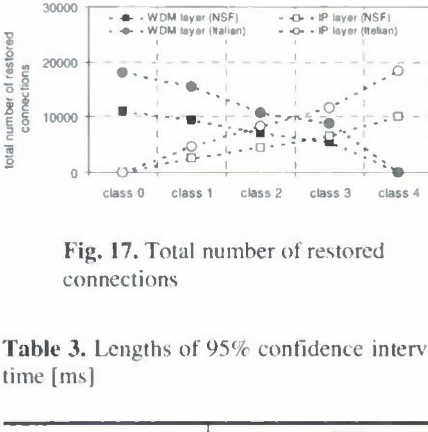


Fig. 17. Total number of restored connections

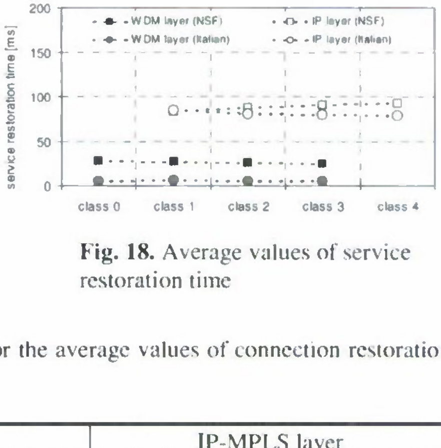


Fig. 18. Average values of service restoration time

Table 3. Lengths of 95% confidence intervals for the average values of connection restoration time [ms]

network	WDM layer					IP-MPLS layer				
	class 0	class 1	class 2	class 3	class 4	class 0	class 1	class 2	class 3	class 4
NSF	2.24	2.94	2.91	3.90	-	-	4.96	2.83	1.92	1.68
Italian	0.24	0.40	0.27	0.00	-	-	0.84	0.70	0.74	0.77

The average values of service recovery time in the WDM layer, shown in Fig. 18, remain at the same low level (typical to the link protection scheme). This is true independent of the service class number, since similar scopes of WDM-layer protection are provided for all the service classes. In each case, the values of service restoration time in the IP-MPLS layer are several times greater than in the WDM layer.

Fig. 19 shows the aggregate values of serviee reeovery time as a funetion of the serviee elass number. Eaeh aggregate value was caleulatcd as the sum of all the values of eonnection restoration time, mcasured in a single simulation seenario. Thesc results give another proof of effieieney of the introdueed approaeh. The ratio between elasses m = 0 and m = 4 was even of order 1:15 (for the Italian Network). Aggregate values of serviee restoration time for the highest serviee elass (m = 0) were always the shortest ones.

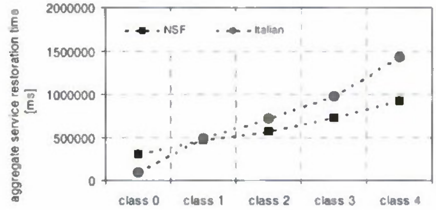


Fig. 19. Aggregate values of service restoration time

4.3 Ratio of Solution Quality Improvement

Fig. 20 shows the advantage of the MLS-GEN-H genetie approaeh over the referenee MLS-H method of solving the survivable routing problem in eaeh network layer exactly once. The ratio of solution quality improvement is given here in terms of the decrease of the total number of WDM link ehannels, required to provide the survivable routing of demands from the set D_{IP}, as a funetion of the population number. For each i-th population, this ratio is given for its best ehromosome.

Duc to maintaining a eertain number (here $|CH|$ = 20) of the so far calculated best chromosomcs for computations in each next itcration and performing the parallcl search in the solution spaee, the proposed MLS-GEN-H genetic algorithm obtained the results up to 22.55% better (Italian Network), eompared to the results of the referenee MLS-H approaeh. In this ease, 632 against initial 816 link ehannels were needed. They also show that, after a reasonable number of iterations (e.g. 400), a good solution may be found and further improvement is hardly possible.

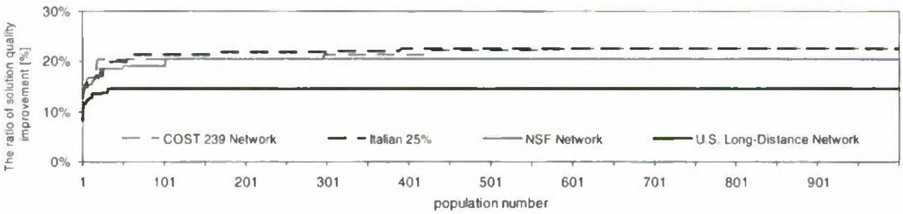


Fig. 20. The ratio of solution quality improvement (network load: 25% node pairs chosen randomly; population size: 20)

Fig. 21 shows the results for the U.S. Long-Distance Network for the ease of varying network load. They are presented for four sizes of demand set D_{IP}, eonsisting of randomly ehosen 25, 50, 75 and 100% of all the network node pairs, aecordingly. They show that the averagc number of iterations needed to obtain a good-quality solution increascs with the inerease of the network load.

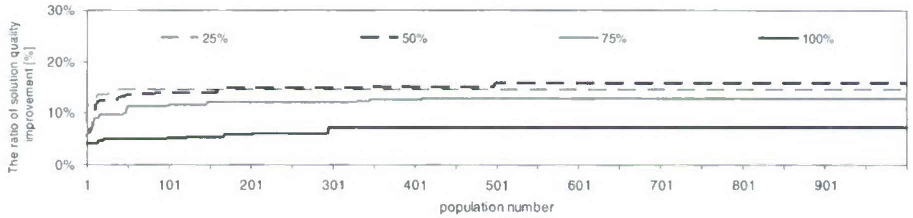


Fig. 21. The ratio of solution quality improvement for U.S. Long-Distance Network (various network loads; population size: $|CH| = 20$)

Table 4. Numbers of required WDM link channels for the best solutions for different population sizes

	U.S. Long-Distance	NSF	COST 239	Italian
best solution for $\|CH\| = 20$ chromosomes/population	1530	256	378	632
best soluiion for $\|CH\| = 60$ chromosomes/populaiion	1596	260	380	554
besi soluiion iieration number (20 chromosomes/population)	72	103	809	535
best solution iteration number (60 chromosomes/population)	30	514	839	806

Table 4 presents the numbers of iterations that were required to find the respeetive best solutions for all the analyzed networks with the demand sets eonsisting of 25% of randomly chosen node pairs. They are shown for two cases of population size, consisting of $|CH| = 20$ and 60 ehromosomes, aeeordingly. In all eases exeept for the Italian Network, after 1000 iterations, the obtained results were worse for the greater population size. This may mean that the number of iterations, needed to get the solution of a given quality, may be greater for larger populations.

5 Conclusions

In this paper we introduced the novel class-based algorithm of survivable routing in IP-MPLS/WDM networks providing differentiated levels of serviee survivability, based on the serviee elass number. This differentiation was defined in terms of the values of serviee recovery time and the frequeney of performing the time-consuming recovery aetions in the IP-MPLS layer.

The original problem of survivable routing in IP-MPLS/WDM network was divided into two subproblems, one for eaeh network layer. Finding the solution to the survivable routing in the IP-MPLS layer was followed by obtaining the results in the WDM layer. However, solving the two subproblems separately in a sequential manner might eertainly lead to the results far from the optimal ones. To overeome this problem, the metaheuristie approaeh, ealled MLS-GEN-H, based on genetic programming was proposed to improve the quality of results by solving the problem iteratively. This in turn enabled to perform the parallel search when finding the best solution. As a result, MLS-GEN-H algorithm achieved the advantage of up to 22.55%, compared to the results of reference MLS-H method of solving the two subproblems in each network layer exactly once.

References

1. Ali, M.: Shareability in Optical Networks: beyond Bandwidth Optimization. IEEE Optical Communications 42(2), 11–15 (2004)
2. Autenrieth, A.: Recovery Time Analysis of Differentiated Resilience in MPLS. In: Proc. Design of Reliable Communication Networks 2003 - DRCN 2003, pp. 333–340 (2003)
3. Bhandari, R.: Survivable Networks: Algorithms of Diverse Routing. Kluwer Academic Publishers, Boston (1999)
4. Bigos, W., et al.: Survivable MPLS over Optical Transport Networks: Cost and Resource Usage Analysis. IEEE J. Select. Areas Commun. 25(5), 949–962 (2007)
5. Cui, X., et al.: Optimization of Multilayer Restoration and Routing in IP-over-WDM Networks. In: Proc. OFC/NFOEC, pp. 1–10 (2008)
6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading (2002)
7. Harle, D., Albarrak, S.: Differentiated Survivability in a Distributed GMPLS-based IP-over-Optical Network. In: Proc. ONDM 2008, pp. 1–5 (2008)
8. Liu, Y., Tipper, D., Vajanapoom, K.: Spare Capacity Allocation in Two-Layer Networks. IEEE J. Select. Areas Commun. 25(5), 974–986 (2007)
9. Molisz, W., Rak, J.: Region Protection/Restoration Scheme in Survivable Networks. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 442–447. Springer, Heidelberg (2005)
10. Molisz, W., Rak, J.: A Novel Class-based Protection Algorithm Providing Fast Service Recovery in IP/WDM. In: Das, A., Pung, H.K., Lee, F.B.S., Wong, L.W.C. (eds.) NETWORKING 2008. LNCS, vol. 4982, pp. 338–345. Springer, Heidelberg (2008)
11. Mukherjee, B.: Optical WDM Networks. Springer, Heidelberg (2006)
12. Qin, Y., Mason, L., Jia, K.: Study on a Joint Multiple Layer Restoration Scheme for IP over-WDM Networks. IEEE Network 17(2), 43–48 (2003)
13. Pickavet, M., et al.: Recovery in Multilayer Optical Networks. IEEE J. of Lightwave Technology 24(1), 122–133 (2006)
14. Ratnam, K., Zhou, L., Gurusamy, M.: Efficient Multi-Layer Operational Strategies for Survivable IP-over-WDM Networks. IEEE J. Select. Areas Commun. 24(8), 16–31 (2006)
15. Sahasrabuddhe, L., Ramamurthy, S., Mukherjee, B.: Fault Management in IP-over-WDM Networks: WDM Protection vs. IP Restoration. IEEE J. Select. Areas Commun. 20(1), 21–33 (2002)
16. Tapolcai, J., et al.: A New Shared Segment Protection Method for Survivable Networks with Guaranteed Recovery Time. IEEE Trans. on Reliability 57(2), 272–282 (2008)
17. Wang, H., Modiano, E., Médard, M.: Partial Path Protection for WDM Networks: End-to-End Recovery Using Local Failure Information. In: Proc. ISCC 2002, pp. 719–725 (2002)
18. Wauters, N., Demeester, P.: Design of the Optical Path Layer in Multiwavelength Cross connected Networks. IEEE J. Select. Areas Comn. 1(5), 881–892 (1996)
19. Xiong, Y., Mason, L.G.: Restoration Strategies and Spare Capacity Requirements in Self healing ATM Networks. IEEE/ACM Trans. on Netw. 7(1), 98–110 (1999)
20. `http://www.pg.gda.pl/~jrak/MMM-ACNS2010WMJR.pdf`

Predictive Security Analysis for Event-Driven Processes

Roland Rieke and Zaharina Stoynova

Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany
{roland.rieke,zaharina.stoynova}@sit.fraunhofer.de

Abstract. This paper presents an approach for predictive security analysis in a business process execution environment. It is based on operational formal models and leverages process and threat analysis and simulation techniques in order to be able to dynamically relate events from different processes and architectural layers and evaluate them with respect to security requirements. Based on this, we present a blueprint of an architecture which can provide decision support by performing dynamic simulation and analysis while considering real-time process changes. It allows for the identification of close-future security-threatening process states and will output a predictive alert for the corresponding violation.

Keywords: predictive security analysis, analysis of business process behaviour, security modelling and simulation, complex event processing.

1 Introduction

With the increased adoption of service oriented infrastructures and architectures, organisations are starting to face the need for an accurate management of cross-process and cross-layer security information and events. The main constraint of current systems is the restriction of Security Information and Event Management (SIEM) [8] to network infrastructure, and the inability to interpret events and incidents from other layers such as the service view, or the business impact view, or on a viewpoint of the service itself. Conversely, specific service or process oriented security mechanisms are usually not aware of attacks that exploit complex interrelations between events on different layers such as physical events (e.g. access to buildings), application level events (e.g. financial transactions), business application monitoring, events in service oriented architectures or events on interfaces to cloud computing applications. Nevertheless, next generation systems should be able to interpret such security-related events with respect to specific security properties required in different processes. On the base of these events, the system should be able to analyse upcoming security threats and violations in order to trigger remediation actions even before the occurrence of possible security incidences.

In this paper we propose to combine process models with security policies and a security model in order to identify potential cross-cutting security issues. We furthermore suggest a blueprint of an architecture for predictive security analysis

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 321–328, 2010.

that leverages process and threat analysis and simulation techniques in order to be able to dynamically relate events from different execution levels, define specific level abstractions and evaluate them with respect to security issues.

2 Related Work

Our work combines aspects of process monitoring, simulation, and analysis. Some of the most relevant contributions from these broad areas are reviewed below.

Business Activity Monitoring (BAM). The goal of BAM applications, as defined by Gartner Inc., is to process events, which are generated from multiple application systems, enterprise service busses or other inter-enterprise sources in real time in order to identify critical business key performance indicators and get a better insight into the business activities and thereby improve the effectiveness of business operations [6]. Recently, runtime monitoring of concurrent distributed systems based on LTL, state-charts, and related formalisms has also received a lot of attention [5,3]. However these works are mainly focused on error detection, e.g. concurrency related bugs. In the context of BAM applications, in addition to these features we propose a *close-future* security analysis which provides information about possible security risks and threats reinforcing the security-related decision support system components.

Complex Event Processing (CEP). CEP provides a powerful analytic computing engine for BAM applications which monitor raw events as well as the real-time decisions made by event scenarios. David Luckham [4] provides us with a framework for thinking about complex events and for designing systems that use such events. A framework for detecting complex event patterns can be found e.g. in [10]. However such frameworks concentrate on detecting events important for statistical aspects, redesign and commercial optimisation of the business process. Here we want to broaden the scope of the analysed event types by introducing *complex security events* in the CEP alphabet.

Simulation. Different categories of tools that are applicable for simulation of event-driven processes including process modelling tools based on different semi-formal or formal methods such as Petri Nets [2] or Event-driven Process Chains (EPC) [1]. Some process managements tools, such as FileNet [7] offer a simulation tool to support the design phase. Also some general purpose simulation tools such as CPNTools [11] were proven to be suitable for simulating business processes. However, independently from the tools and methods used, such simulation tools concentrate on statistical aspects, redesign and commercial optimization of the business process. On the contrary, we propose an approach for *on-the-fly* intensive dynamic simulation and analysis considering the current process state and the event information combined with the corresponding steps in the process model.

Security Information Management (SIM). SIM systems generally represent a centralized server acting as a "security console", sending it information about security-related events, which displays reports, charts, and graphs of that information, often in real time. Commercial SIEM products include Cisco Security Monitoring Analysis and Response System (`http://www.cisco.com/en/US/products/ps6241/index.html`), EventTracker by Prism Microsystems (`http://www.prismmicrosys.com/EventTrackerSIEM/index.php`), SenSage (`http://www.sensage.com/products/sensage-40.php`) and others. All these products monitor the low-level events (such as network events) and perform event correlation only on the base of event patterns and rules. Our approach additionally considers the business process level events combined with the current process state and business process information provided by a process specification.

3 Blueprint of Architecture for Security Event Processing and Predictive Security Monitoring

In this section we introduce our approach for security evaluation of event-driven processes. Figure 1 depicts the core components which we consider necessary in order to be able to perform a security event processing and monitoring analysis in the context of a running event-driven business process.

The input elements which we need comprise, (1) a *process model* given in a notation such as EPC, BPEL, YAWL or BPMN that contains a specification of the events which can be triggered during runtime, (2) *security policies* which contain information about the relations between the users involved in the process, their roles and the relations between the roles and resources deployed by the process, (3) a *security model* that should provide information about the process's

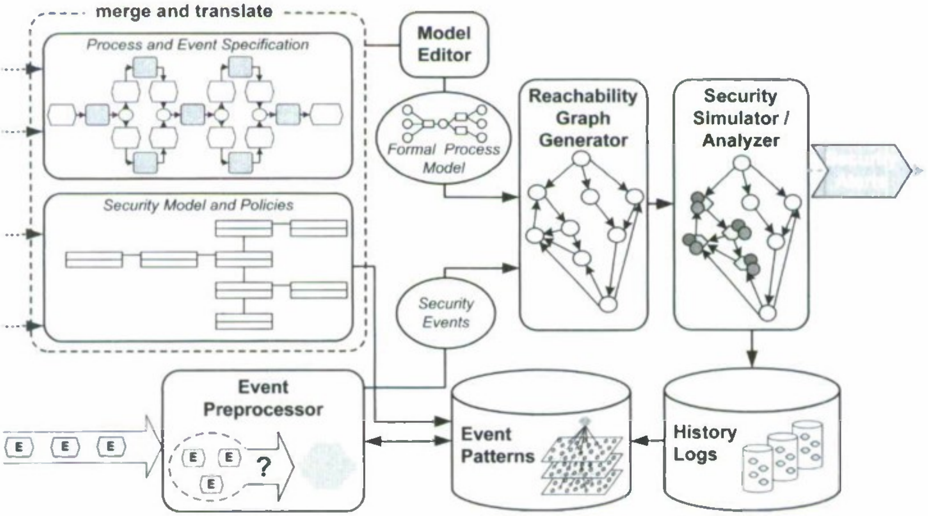


Fig. 1. Predictive Security Analyser

predefined security requirements which will be used to construct the security events patterns, and, (4) *real-time events* which will be triggered during runtime.

Model Editor. In order to analyse the system behaviour with tool support, an appropriate formal representation has to be chosen because semi-formal languages such as BPMN allow to create models with semantic errors [2]. In our approach, we use an operational finite state model based on *Asynchronous Product Automata (APA)* [9]. An APA consists of a family of so called *elementary automata* communicating by common components of their state (shared memory). The process model, the organisational model and the security model should be imported and merged in a high-level model of the process and then this model is translated into an APA, which will enable the computation of the possible system behaviour. In general, we could also use other descriptions of processes with unambiguous formal semantics here such as the approaches in [2] for BPMN or [1] for EPC that allow for computation of possible system's behaviour.

Reachability Graph Generator. Formally, the behaviour of an APA can be given by a reachability graph which represents all possible coherent sequences of state transitions starting with the initial state. In the context of on-the-fly security analysis the reachability graph will represent the path given by the already triggered events, forwarded by the Event Preprocessor. The computation will be automatically paused each time when the current state (according to the triggered events) of the process is reached. In the context of predictive simulation analysis the Reachability Graph Generator computes all possible near-future paths according to the given process specification, (e.g. sequences of at most 2-3 plausible events). This will allow exhaustive analysis of all near-future states to be performed in order to compute whether there exist possible security-threatening states of the process which can compromise the process security and match some of the event patterns saved in the Event Patterns database.

Security Simulator/Analyser. During the computation of the graph this component will check for each state, whether the specified security properties are fulfilled and trigger security alarms when possible security violations are found. Furthermore, it is possible to detect new security violations that were not predicted by the available security patterns. In order to include them in the analysis of future process instances, they will be logged in the History Logs database and then they will be transformed into security event patterns and saved in the Event Patterns database. The simulator will also enable security analysis by performing intensive simulation which inspects the behaviour of complex/parallel processes under given hypotheses (*what-if analysis*) concerning changes in the organisational model/security policies or the process model.

Security Event Patterns. These patterns which are relevant for the corresponding process are kept in the Event Patterns database and they should be extracted from the provided security model. In order to be able to reason about potential security problems, based on real life events, specific abstractions are included in this extraction process so that the abstraction levels for the various types of security-related events can be interrelated. Solutions for these kind of security analysis are already available but usually limited to a narrow field of application such as IDS where e.g. the detection of a number of abnormal connections could lead to a "worm detection" alarm. We propose a generic approach leveraging these ideas and incorporating other types of security related events.

Event Preprocessor. In the context of on-the-fly security analysis the Event Preprocessor is responsible for receiving the real-life events triggered during runtime, matching them against the available security event patterns and forwarding them to the Reachability Graph Generator. During predictive security analysis the Event Preprocessor will generate all possible events according to the process specification and will match them against the event patterns. Then it will forward them to the Reachability Graph Generator in oder to enable the computation of the process graph.

History Logs. In the History Logs database newly detected security-violating sequences of events will be logged. These will be used to create new security event patterns.

4 An Application Scenario

For illustrating how our architecture components, described in the previous section, collaborate we will refer to a common example scenario for online credit application.

4.1 Process Model

In an EPC graph events are represented as hexagons and functions that describe state transitions are represented as rounded rectangles. Now consider the online credit application process expressed in EPC notation in Fig. 2. The process starts when an applicant submits an application form. Upon receiving a new application form a credit clerk performs checks in order to validate the applicant's income and other relevant information. Depending on the requested loan amount different checks are performed. Then the validated application passed on to a manager to decide whether to accept or reject it. In both cases the applicant is notified of the decision and the process ends.

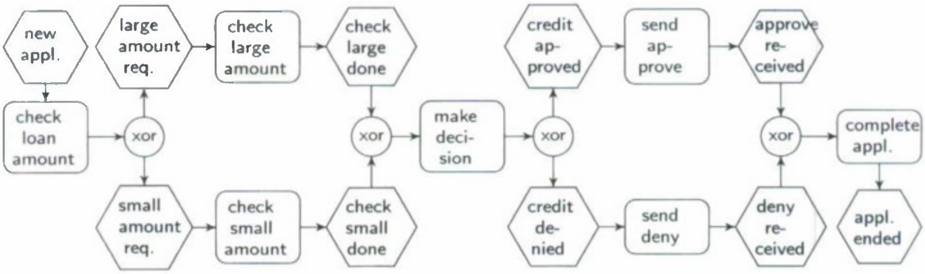
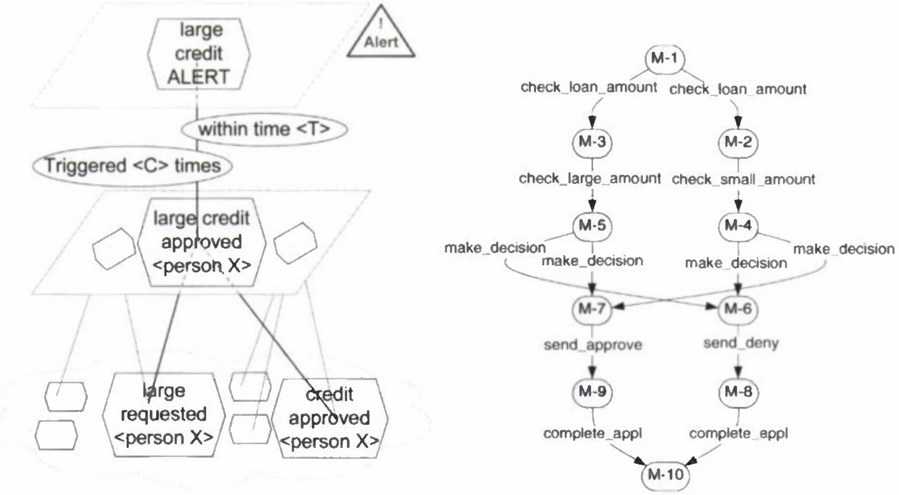


Fig. 2. Business Process Model



(a) On-the-fly security violations (b) Reachability graph of business process

Fig. 3. Predict near future security violations

4.2 Predicting Security Events

In our example scenario we consider the security event "*large credit ALERT*" which is raised when too many large credits are approved for one customer (see Fig. 3(a)). This is an example of an event abstraction or complex event generated by a certain sequence of simple events, triggered in the process. Such complex events are generated by CEP engines whenever certain predefined sequences of events have been triggered.

Additionally, we apply such complex event patterns in a predictive way. This means that whenever an event pattern is *probably* going to match by taking into account a current partial match and a possible continuation of the current state, these abstractions can be generated prior to the real-time triggering of the simple events. In our example we generate an abstraction of the atomic events "*large amount requested*" and "*credit approved*" triggered by the same customer, namely the complex event "*large credit approved*". Then if this complex event is generated e.g. two times within a certain time and according to security regulations only two large credits can be given to one customer we can generate the alert "*large credit ALERT*" in the upper abstraction level prior to the next approval in order to ensure that the security regulations will not be overseen by taking the credit decision.

4.3 Operational Model for Security Event Prediction

A computation of the possible system behaviour of a formal APA model of the business process in Fig. 2 results in the reachability-graph depicted in Fig. 3(b). The state M-3 e.g. represents the situation where an event of type "*large amount* *requested*" is available and can be processed by the action "*check_large_amount*" which in turn will trigger an event of type "*check large done*". After this, the process is in state M-5, where the action "*make_decision*" can be executed and lead to one of the two possible followup states M-6 or M-7. M-7 is reached iff the decision results in an event "*credit approved*".

From this we now conclude that a predictive alert "*large credit ALERT*" can be generated if, (1) the system is in a state where the number of large credits allowed for one customer is exhausted, (2) an event "*large amount requested*" for the same customer is received, and, (3) an evaluation of possible continuations of the process's behaviour based on the operational model shows that an additional event of type "*large credit approved*" is possible within the forecast window.

The method described in this paper addresses security properties that can be stated as safety properties. Possible violations of these properties are identified by reachable states in the predicted system behaviour. Some examples of security related event types that can be analysed by the method given in this paper are:

Confidentiality. Consider an event sending a cleartext password. Predict that in one possible continuation of a process, an event about processing a cleartext password locally may lead to an event sending that password.

Authenticity. Consider the physical presentation of a token which is known to be unique such as a credit card or passport as parameter of two different events with very close time and very different location.

Authorisation. Consider two events with persons with the same biometric parameters in different locations at the same time.

Integrity/Product counterfeiting. Consider RFIDs being scanned in places where they are not expected.

Integrity/Safety. Consider two trains on the same railtrack. Predict that a specific constellation of switches leads to a crash in one possible continuation.

5 Conclusions and Further Work

In this paper we proposed a blueprint of an architecture for predictive security analysis of event-driven processes that enables exhaustive process analysis during runtime based on the triggered real-life events. Our approach is based on the specification of an operational finite state model of the process behaviour We have demonstrated how our methods can be applied in order to ensure certain security regulations in the process of online credit application and how we can construct event abstractions on different levels in order to detect current and near-future threats.

Currently our components are prototypically implemented without automated merging and translation mechanisms for the input models and specifications, automated event pattern extraction and new event pattern composition. We used the *SH verification tool* [9] to analyse an exemplary business process model for different concrete instantiations (numbers of clients, and time-horizon) of the model. In the future, we will further develop such techniques in order to automate the security analysis and simulation and extend the method to cover liveness properties.

Furthermore, alerts in today's monitoring systems by themselves bring little value in the process security management if they cannot be acted upon. Therefore, we have to provide additionally to the alerts alternative counter-measure scenarios that can be quantifiable evaluated thanks to simulation. In this way our analysis can be extended to provide feedback to the operators on feasibility and impacts of both attacks and counter-measures.

Acknowledgments. The work presented in this paper was developed in the context of the project Alliance Digital Product Flow (ADiWa) that is funded by the German Federal Ministry of Education and Research. Support code: 01IA08006F.

References

1. Dijkman, R.M.: Diagnosing Differences Between Business Process Models. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 261–277. Springer, Heidelberg (2008)
2. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. 50(12), 1281–1294 (2008)
3. Kazhamiakin, R., Pistore, M., Santuari, L.: Analysis of communication models in web service compositions. In: WWW 2006: Proc. of the 15th International Conference on World Wide Web, pp. 267–276. ACM, New York (2006)
4. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley, Reading (2002)
5. Massart, T., Meuter, C.: Efficient online monitoring of LTL properties for asynchronous distributed systems. Tech. rep., Université Libre de Bruxelles (2006)
6. McCoy, D.W.: Business Activity Monitoring: Calm Before the Storm. Gartner Research (2002)
7. Netjes, M., Reijers, H., Van der Aalst, W.P.: Supporting the BPM life-cycle with FileNet. In: Proceedings of the Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2006), held in conjunction with the 18th Conference on Advanced Information Systems (CAiSE 2006), Luxembourg, pp. 497–508. Namur University Press, Namur (2006)
8. Nicolett, M., Kavanagh, K.M.: Magic Quadrant for Security Information and Event Management. Gartner RAS Core Reasearch Note (May 2009)
9. Ochsenschläger, P., Repp, J., Rieke, R., Nitsche, U.: The SH-Verification Tool Abstraction-Based Verification of Co-operating Systems. Formal Aspects of Computing, The International Journal of Formal Method 11, 1–24 (1999)
10. Pietzuch, P.R., Shand, B., Bacon, J.: A framework for event composition in distributed systems. In: Endler, M., Schmidt, D.C. (eds.) Middleware 2003. LNCS, vol. 2672, pp. 62–82. Springer, Heidelberg (2003)
11. Rozinat, A., Wynn, M.T., van der Aalst, W.M.P., ter Hofstede, A.H.M., Fidge, C.J.: Workflow simulation for operational decision support. Data Knowl. Eng. 68(9), 834–850 (2009)

Virtual Environment Security Modeling

Dmitry Zegzhda and Ekaterina Rudina

Saint-Petersburg State Polytechnical University,
Polytekhnicheskaya str. 29, Saint-Petersburg, Russian Federation
{dmitry,e-katerina}@ssl.stu.neva.ru,
ekaterina.rudina@gmail.com

Abstract. Virtualization allows to manage a lot of properties of computer systems including the secnrity of information processing. Goal of this investigation is to state conditions of the ability of virtualization mechanism to guarantee satisfying of the security policy. It is formally proved that if the virtual environment is untrusted, virtualization mechanism should be run on the trusted operating system.

Keywords: virtualization, hypervisor, virtual environment, trusted system, information security, untrusted application, security modeling.

1 Introduction

In general virtualization is a technique of the computer resources representation to obtain new properties of these resources use.

The formal requirements for virtualizable architectures were originally defined by Popek and Goldberg [1]. Their virtual machine monitor was built using the call interception technique. Binary translation code has become highly competitive to this approach nowadays [2]. However, the concepts of the virtualization and virtual machine monitor (hypervisor) are being defined and widely used in this paper independently of the method of their realization.

Hypervisor should answer at least the two conditions. Firstly, execution of the virtual environment should be invariable to the execution of the non-virtual system (equivalence property). Secondly, the virtualized resources should be completely separated (resource control property). Efficiency property is outside of this investigation.

A hypervisor usually works as a regular application controlled by a (usually non-virtual) system and does not prevent using other applications at the same time. Virualization technique commonly is inessential. If defined so, hypervisors of both Type-I and Type-II will be considered.

2 Related Works

A tendency has recently been observed to virtualize the information processing means in order to enhance their security. Different approaches to construct the

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 329–336, 2010.

processing architecture has been used, most of them being aimed at the data isolation and/or virtual separation of the processes. Examples of such solutions can be found in [3], [4], [5], [6].

In [3] is asserted that the operating system and applications currently running on a real machine should relocate into a virtual environment, because it enables services to be added below the operating system and to do so without trusting or modifying the operating system or applications. Three services are demonstrated as an examples (secure logging, intrusion prevention and detection, and environment migration), but the formal substantiation of that approach is absent.

In [4] is presented an architecture that retains the visibility of a host-based intrusion detection system, but pulls the IDS outside of the host for greater attack resistance.

The authors of [5] use the technique, that is analogous to the virtualization, to isolate the effects of untrusted program execution from the rest of the system. Isolation is achieved by intercepting and redirecting file modification operations made by the untrusted process so that they access a "modification cache" invisible to other processes in the system. Key benefits of this approach are that it requires no changes to the untrusted programs (to be isolated) or the underlying operating system; it cannot be subverted by malicious programs; and it achieves these benefits with acceptable runtime overheads. It is the same benefits that offers the virtualization.

In [6] is also offered the approach to program isolation. The dangerous system calls are intercepted and filtered via the Solaris process tracing facility. The declared advantage is to reduce the risk of a security breach by restricting the program's access to the operating system. That access can be also restricted by a virtualization hypervisor (according to a resource control property).

Nevertheless, the virtualization technology is able to provide more opportunities to secure information processing than simple isolation or access restrictions.

3 Problem Definition

In this paper we will regard program virtualization for executing untrusted applications in secure environment. Programs which cannot be directly executed in this environment may be run by a virtual machine, though the operating system is to control the access of the applications to the data which are to be kept secure. Better functioning of the operating system can be achieved to a less cost in this situation.

The problem is to define properties of the physical system which will be inherited by the virtual environment. To describe the invariable property of programs [1] a number of statuses of the virtualized system is considered as a homomorphic inage of the statuses of a real system. Which properties of the virtualized system does the homomorphism retain? Under what conditions can a virtual system inherit not only the properties of the virtualized system, but of the base system as well? To answer these questions we need to:

- specify which security aspects can formally be guaranteed using the approach;
- simulate a model of a computer, which can execute a definite set of programs;
- specify the hypervisor's properties in terms of this simulation and according to the definition given in [1] and describe how the initial system relates to its virtual image, executed under the hypervisor's control.

The goal of this investigation is to formulate the conditions under which any application executing in virtualized system environment keeps secure, while executing the same applications in the same but not virtualized environment may not be secure.

4 The Computing System Model. Integrated Security Condition

Let us describe a model of a computing system M on which the problem definition will be based.

The crucial feature of the model is resources typification. A resource in computing system is not only a named object keeping data. It is characterized by its own access techniques and its interpretation when obtaining the data from it. From the virtualization mechanism viewpoint, different computing system resources should be regarded according to the technique of their interpretation and to whether they can be virtualized using this mechanism separately from other resources. In other words, the typification of these resources in terms of their virtualization possibility should be applied. The resource type is constant throughout its lifetime. Resources of different types can keep or transmit the same data. The single-valued identification of the resource is necessary for its obtaining or modification.

Thus, each uniquely identified resource of a computing system is determined by its type and the data it keeps at every particular moment of time.

The model $M = (P, R, TR, D, \tau, \delta, F, Prg, \varphi)$ is specified by the following sets and mappings:

P a set of subjects (processes)
$R,\ P \subseteq R$ a set of objects (resources)
TR a finite set of resource types
D the data kept or transmitted by the resource
$\tau : R \to TR$ resource type function
$\delta : R \to D$ the function of data kept/transmitted by the resource
$F = \{f_i\},\ i \in 1 : n$ a finite set of functions defining the system status according to which resources it was applied
$\forall f_i \exists dep_i = \{tr_{ij}\}$ resources the function executing depends on
$\forall f_i \exists eff_i = \{tr_{ik}\}$ resources the function executing tells on
F^* a set of finite function sequences from F
$Prg \subset F^*$ a set of programs for the system
$\varphi : P \to Prg$ a function matching a program to each process

$S = (P, R, p)$ a sequence describing the system status at each moment
$p \in P$ an active process
$C = \{S\}$ a set of system statuses

Program behavior $prg \in Prg$ is described by a sequence $B_{prg} = (I_{prg}, O_{prg}, A_{prg})$, where

$I_{prg} \subseteq R$ a set of incoming resources
$O_{prg} \subseteq R$ a set of outcoming resources
$A_{prg} : \delta(I_{prg}) \to \delta(O_{prg})$ an algorithm of program performance

Let us consider such security aspects as confidentiality of the data processed, accessibility of this data (given that the access requested is legitimate), data and environment integrity. To control the security in one of these aspects some formal criterion is necessary. The criterion can be described as a predicate VER, active in a set of data. The predicate VER performance should be guaranteed at least for a specified subset of the computing system. The types of sensitive resources build a subset $CR \subseteq TR$. The security condition may then be put like this:

$$(\forall r \in R(\tau(r) \in CR) \Rightarrow VER(\delta(r))) \tag{1}$$

This condition may be reduced to the requirement of confidence or integrity of some data, to some special requirements to computer resources during processing of these resources and so on. Hence, using of integrated security condition allows us to apply our approach more broadly.

5 Virtual System Properties

Let us analyze the properties of the architecture considered using the model we have introduced.

There is an insecure general-purpose system A and a secure system H. The systems are used for processing data, kept or transmitted by other resources of different types. The sets of their resource types completely or partially coincide. The systems A and H are described by a model of a computing system M. There is a predicate that describes executing the security aspect $VER : D \to \{true, false\}$ and type (1) data security condition. Data representation in the both systems A and H is identical, i.e. their sets D coincide. As a result, predicate VER definition in systems A and H is identical.

System A, described by the model $M^A = (P^A, R^A, TR^A, D, \tau^A, \delta, F, Prg^A, \varphi)$, may be virtualized under system H control as the third system V, described by a model $M^V = (P^V, R^V, TR^V, D, \tau^V, \delta, F, Prg^V, \varphi)$

1. The following properties are fulfilled for the system A relative to the system V:
   (a) $R^A \subseteq R^V$
   (b) Invariability of the virtual environment relative to the real machine
   $\exists H_S : C^A \to C^V$ which is such that
   $\forall f \in F, \forall S^A \in C^A \Rightarrow \exists S^V \in C^V, \exists f^* \in F^* : (H(f(S^A)) = f^*(H(S^A)))$

   (c) Hypervisor control over virtualization of the resources of types $VR \subseteq TR^A$
   $\forall f_i : dep_i \subseteq VR \vee eff_i \subseteq VR$
   $\forall S_1^A \in C^A, S_2^A = f_i(S_1^A), S_1^V = H(S_1^A), S_2^V = H(S_2^A)$
   $\exists f_1, f_2, \ldots, f_k :$
   $(S_2^V = f_1 f_2 \ldots f_k(S_1^V) \wedge$
   $(j \in 1 : k$
   $(dep_j \cap VR = \emptyset \vee eff_j \cap VR = \emptyset) \vee$
   $(S_j^V = f_1 f_2 \ldots f_j(S_1^V = (P_x, R_x, p_x) \Rightarrow \varphi(p_x) = VH))$

2. For the base system H, described by the model, are fulfilled the following properties:
   (a) $P^H \subseteq P^V$
   (b) $R^H \subseteq R^V$
   (c) $TR^H \subseteq TR^V, \forall r \in R^H \Rightarrow \tau^V(r) = \tau^H(r)$

3. The sets of programs, valid in A, H and V, relate to each other as $Prg^V = Prg^A \cup Prg^H$ and $\forall p \in P^H \Rightarrow \varphi^V(r) = \varphi^H(r)$ is met.

4. Resources, used for keeping and/or transmitting the sensitive data, are subsets of the virtualized resources:
   $CR^A \subseteq VR$.

5. Behavior of the virtualization hypervisor as a program in system H
   $B_{VH} = (I_{VH}, O_{VH}, A_{VH})$, $I_{VH} \subseteq R$, $O_{VH} \subseteq R$, $A_{VH} : \delta(I_{VH}) \rightarrow \delta(O_{VH})$
   answers the security condition 1
   $\forall S_1^H \in C^H, \forall f_i \in F :$
   $(S_1^H = (P_1^H, R_1^H, p_1^H) \wedge f_i(S_1^H) = S_2^H = (P_2^H, R_2^H, p_2^H) \wedge \varphi(p_1^H) = VH) \Rightarrow$
   $(\forall r \in R_1^H : (\tau^H(r) \in CR^H \Rightarrow VER(\delta(r))) \Rightarrow$
   $(\forall r \in I_{VH} \cup O_{VH} \subseteq R_2^H(\tau^H(r) \in CR^H \Rightarrow VER(\delta(r))))$

We will consider this conditions being true for the next propositions. Let's view what we should demand under these conditions if we want to supply the secure execution of untrusted applications.

6 Security Requirements in the Virtual System

The following statement is proved.

Theorem 1. Let the following conditions for the resource typification functions be met in the systems A and V

the resource typification function is mapped from M^A to M^V homomorphically, i.e. $\exists \chi : TR^A \rightarrow TR^V, \forall r \in R^A \subseteq R^V(\tau^V(r) = \chi(\tau^A(r)))$

and $\forall t \in TR^A : t \in CR^A \Leftrightarrow \chi(t) \in CR^H$

The virtual system V meeting the given conditions provides any program of system A secure execution.

In other words, when the given conditions are met, any program's behavior in V will be changed by the virtualization hypervisor so as to meet the security requirements, even if the behavior of this program in A is insecure.

Proof. The proof of this statement is being made by reduction to absurdity. If this statement is false, predicate VER will be met and not be met simultaneously.

Nonsecure program behaviour in the system A means that some function in some state of this system violates the security condition:

$$\begin{gathered}\exists S_1^A \in C^A, \exists f_i \in F: \\ (S_1^A = (P_1^A, R_1^A, p_1^A) \wedge f_i(S_1^A) = S_2^A = (P_2^A, R_2^A, p_2^A) \wedge \phi(p_1^A) = prg) \Rightarrow \\ \forall r \in R_1^A : (\tau(r) \in CR^A \Rightarrow VER(\delta(r)) = true) \wedge (\exists r \in O_{2prg}^A \subseteq R_2^A : \tau(r) \in CR^A \wedge VER(\delta(r)) \neq true).\end{gathered}$$

Particularly, that is followed by $eff_i \cap CR^A \neq \emptyset$.

Let us consider that if program were run under virtual machine, its behavior wouldn't change. Then following condition is met:

$$\begin{gathered}\exists S_1^V, S_k^V, \exists f_1, f_2 \dots f_k : S_1^V = H(S_1^A), \\ S_k^V = H(S_2^A) = H(f_i(S_1^A)) = f_1 f_2 \dots f_k(H(S_1^A)).\end{gathered}$$

Homomorphism describing invariability of the virtual environment relatively to real environment saves the data representation. Resource type function according to considered condition is mapped from real environment onto virtual environment also homomorphically:

$$\exists \chi : TR^A \rightarrow TR^V, \forall r \in R^A \subseteq R^V(\tau^V(r) = \chi(\tau^A(r))).$$
If $TR^H \subseteq TR^V$ then $CR^H \subseteq TR^V$.

Considering that $\forall t \in CR^A \chi(t) \in CR^H$, let's view homomorphic mapping of nonsecure behavior of the function f_i, then

$$\begin{gathered}\exists S_1^V \in C^V, \exists f \in F_i \\ (S_1^V = (P_1^V, R_1^V, p_1^V) \wedge f_1 \dots f_k(S_1^V) = S_k^V = (P_k^V, R_k^V, p_k^V) \wedge \phi(p_1^V) = prg) \Rightarrow \\ ((\forall r \in R_1^V : \tau^V(r) = \chi(\tau^A(r)) \in CR^H \Rightarrow VER(\delta(r)) = true) \wedge \\ \wedge(\exists r \in R_k^V : \tau^V(r) = \chi(\tau^A(r)) \in CR^H \wedge VER(\delta(r)) \neq true)).\end{gathered}$$

So it exists a nonsecure state reached from secure state of the system$V$. (2)

From resource control condition we obtain

$$\begin{gathered}\forall f_i : dep_i \subseteq VR \vee eff_i \subseteq VR \\ \forall S_1 \in C, S_2 = f_i(S_1), S_1^V = H(S_1), S_2^V = H(S_2) \\ \exists f_1, f_2 \dots f_k : (S_k^V = f_1 f_2 \dots f_k(S_1^V)) \vee \\ (j \in 1 \dots k \\ (dep_j \cap VR = \emptyset \wedge eff_j \cap VR = \emptyset) \vee\end{gathered}$$

$$\vee(S_j^V = f_1 f_2 \ldots f_k(S_1^V) = (P_j^V, R_j^V, p_j^V) \Rightarrow \phi(p_{j-1}^V) = VH)).$$

VH behavior is secure in system H:

$$\begin{array}{l} \forall S_1^H \in C^H, \forall f_i \in F : \\ (S_1^H = (P_1^H, R_1^H, p_1^H) \wedge f_i(S_1^H) = S_2^H = (P_2^H, R_2^H, p_2^H) \wedge \phi(p_1^H) = VH) \Rightarrow \\ (\forall r \in R_1^H : (\tau^H(r) \in CR^H \Rightarrow VER(\delta(r)) = true)) \Rightarrow \\ (\forall r \in I_{VH} \cup O_{VH} \subseteq R_2^H(\tau^H(r) \in CR^H \Rightarrow VER(\delta(r)) = true)). \end{array}$$

As next conditions

$$\begin{array}{l} P^H \subseteq P^V, R^H \subseteq R^V, TR^H \subseteq TR^V \\ \forall r \in R^H \Rightarrow \tau^V(r) = \tau^H(r) \\ \forall p \in P^H \Rightarrow \phi^V(p) = \phi^H(p) \end{array}$$

are met, so VH behavior is secure in system V relative to base system resources subset:

$$\begin{array}{l} \forall S_1^V \in C^V, \forall f_i \in F : \\ (S_1^V = (P_1^V, R_1^V, p_1^V) \wedge f_i(S_1^V) = S_2^V = (P_2^V, R_2^V, p_2^V) \wedge \phi(p_1^V) = VH) \Rightarrow \\ (\forall r \in R_1^H \subseteq R_1^V : (\tau^V(r) \in CR^H \Rightarrow VER(\delta(r)) = true)) \Rightarrow \\ (\forall r \in R_2^H \subseteq R_2^V(\tau^V(r) \in CR^H \Rightarrow VER(\delta(r)) = true)). \end{array}$$

Let's show for the considering conditions that the running of the functions $f_i : (dep_i \cap VR = \emptyset \wedge eff_i \cap VR = \emptyset)$ doesn't affect the virtual environment state S_j^V security, where S_j^V is reached with applying one or more of these functions to some secure state S_{j-1}^V. Security condition here is meant relative to base system resources subset:

$$\begin{array}{l} \forall t \in dep_i \cup eff_i(\chi(t) \in CR^H \Leftrightarrow t \in CR^A) \\ (dep_i \cap VR = \emptyset \wedge eff_i \cap VR = \emptyset) \Rightarrow (dep_i \cap CR^A = \emptyset \wedge eff_i \cap CR^A = \emptyset). \end{array}$$

That is followed by $\nexists t \in dep_i \cup eff_i : \chi(t) \in CR^H$.

The subset of resources affected by these functions in virtual environment, doesn't contain resources with types from CR^H subset. Thus, security condition can't be violated by these functions being run in virtual environment.

Hence with the VH behavior security condition relative to resources of the types of CR^Hsubset and the resource control condition for the sequence of states

$j \in 1 \ldots k(S_j^V = (P_j, R_j, VH))$ it follows that

$$\forall r \in R_j^V(\tau^V(r) \in CR^H \Rightarrow VER(\delta(r)) = true).$$

So we obtain

$$\begin{aligned}&\forall f_i : dep_i \cup VR \neq \emptyset \vee eff_i \cup VR \neq \emptyset\\&\forall S_1 \in C, S_2 = f_i(S_1), S_1^V = H^{-1}(S_1), S_2^V = H^{-1}(S_2)\\&\exists f_1, f_2 \ldots f_k : (S_k^V = f_1 f_2 \ldots f_k(S_1^V))\wedge\\&\qquad\qquad \wedge(j \in 1 \ldots k, \forall r \in R_j^V(\tau^V(r) \in CR^H \Rightarrow VER(\delta(r)) = true)).\end{aligned}$$

$$\text{Every state reached from secure state of the system } V \text{ is secure.} \tag{3}$$

Obtained contradiction between (2) and (3) is followed by the absurdity of made consideration. Hence, the behavior of the program being run in virtual environment will be changed, Q.E.D.

7 Conclusion

It is proved that under defined conditions the properties of secure data processing will be inherited by virtual environment. An approach to extend functional capabilities of secure operation systems using the virtualization technique has been formally substantiated. The results obtained can be used to create requirements to the virtual environments adopted in this area. Such approach makes the virtualization mechanism worthwhile in securing not only as a means of data isolation and separation of their processing. It can also provide the inheritance of secure base system's properties by a virtual system.

References

1. Popek, G.J., Goldberg, R.P.: Formal Requirements for Virtualizable Third Generation Architectures. Association for Computing Machinery, Inc. 17(7), 412–421 (1974)
2. Adams, K., Agesen, O.: A Comparison of Software and Hardware Techniques for x86 Virtualization. VMware (2006), `http://www.vmware.com/pdf/asplos235_adams.pdf`
3. Chen, P.M., Noble, B.D.: When Virtual Is Better Than Real. In: 8th Workshop on Hot Topics in Operating Systems, HotOS (2001)
4. Garfinkel, T., Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection. In: Proc. Network and Distributed Systems Security Symposium (2003)
5. Zhenkai, L., Venkatakrishnan, V.N., Sekar, R.: Isolated Program Execution: An Application Transparent Approach for Executing Untrusted Programs. In: Proceedings of Annual Computer Security Applications Conference (2003)
6. Goldberg, I., Wagner, D., Thomas, R., Brewer, E.A.: Secure Environment for Untrusted Helper Applications. In: Proceedings of the 6th Usenix Security Symposium (1996)

Clarifying Integrity Control at the Trusted Information Environment

Dmitry P. Zegzhda, Peter D. Zegzhda, and Maxim O. Kalinin

Information Security Center, St. Petersburg Polytechnical University, St. Petersburg, Russia
{dmitry,zeg,max}@ssl.stu.neva.ru

Abstract. The paper addresses to the technique of integrity control based on security settings evaluation which is made over variable software components. There are formal foundations of integrity control related to finding security settings which form trusted security environment. It also uses iterative search for security settings which are compatible and agreed with each other. Our approach results to a schema of Security and Integrity Control System that combines principles of automated control system and security management.

Keywords: access control, automation, integrity, trusted information environment, security management, security settings.

1 Introduction

Contemporary security claims in IT-systems which are targeted at critical information utilization are that they have to be trusted in reference to information environment (IE) and security components. Basically, IE is a convergence of system software (e.g. operating system) and user applications (e.g., office-related software). To be a *trusted* one in security sense, the IE is to integrate a set of security mechanisms, which should realize protection methods. For the trusted IE (TIE), there is a great desire to accomplish a full set of security aspects, i.e. confidentiality, accessibility, and integrity. Having been emphasized at confidentiality and accessibility, access control methods seems to supply the IE with fair security. But access restrictions are not enough for pure assurance of complete information safety. Concerning with integrity, it is necessary to clarify a traditional definition of overall system integrity and include a set of integrity-related aspects corresponding to reliability of system security components, stability of security settings, and invariance of security regulations. These aspects hurt system security, because they are focused on the TIE's unpredictable properties. We call this problem as '*the problem of integrity*'. This paper proposes a technology targeted to settle it.

2 Background and Related Works

Integrity means assurance that information is authentic and complete [1]. In that sense, integrity problem could be resolved with well-known cryptographic approaches

I. Kotenko and V. Skormin (Eds.): MMM-ACNS 2010, LNCS 6258, pp. 337–344, 2010.

(hash, checksums, etc.). Traditional definition of integrity is based on the data level and it does not involve the system wholeness. As the result, we can not completely trust the IE security that obtains assurance for data but not for the system itself. But demand for system integrity control is raised with two significant reasons:

– *a huge number of vulnerabilities in operating systems and program applications.* Security flaws make the TIE's characteristics and behavior totally unpredictable and instable. As the result, the TIE can not be considered as the reliable (and secure) one. It means that IE is just a system with some number of security properties which depend on security of the components;

– *complexity of modern IT-solutions.* Nowadays, IT-solutions integrate different software products shipped by different vendors. Some software has license limitations on code distribution; therefore there is no possibility to inspect overall system reliability by code analyses.

To solve the first problem, a number of security enforcing IT-solutions was implemented: trusted versions of operating systems (e.g., Trusted Solaris [2], secure editions of UNIX systems [3]); security packs and midware (e.g. RSBAC [4], GRSecurity [5]); security gateways (e.g. Astaro Security Gateway [6]); delegating technologies (e.g., Multiple Independent Levels of Security [7]). All these solutions are united with a principle of system isolation. In that case, TIE stability might be treated as a desired security, but influence of human factor can not provide TIE with any stable integrity. Moreover, the security providing software causes compatibility problems with each other. The second factor means that the checksums calculated for any security component do not guarantee integrity of the whole system. This is a result of 'a system property': summary of the given elementary properties does not directly provide the system with the same property (e.g., correct checksums calculated for executables do not mean the system integrity because binaries can run in different executive environments with different settings which can be changed while the system works).

The TIE with unpredictable properties (integrity as well) can not be treated as pure secured and trusted. System integrity issues (e.g., stability of configuration, invariance of security restrictions) are not resolvable with cryptography techniques and thus they force us to look for new approaches. This paper proposes a technology targeted at solving the problem of integrity on the system level. Soul of the solution is formed with monitoring and controlling of IE's security states with giving a more precise definition for integrity as for a security property.

3 Information Environment Integrity

Traditionally, integrity is the ensuring that information can be relied upon to be sufficiently accurate for its purpose. Term 'integrity' is frequently used when considering IT-security as it is represents one of the primary indicators of security (or lack of it). As mentioned above, integrity is not only whether data are 'right', but whether they are trusted and relied upon. Unfortunately, for system complexity (e.g., either different security components or components with different security) and configurations instability, it is necessary to clarify the integrity definition taking into account all of the system components besides data. We suggest updating the term of integrity. *Integrity* is the ensuring that information environment is stable (invariable). The term 'integrity' is thus transferred from static to dynamic sense. Stable and variable parts of system integrity arc presented in Fig. 1. Stable components include the functional modules that are founded at system designing and building: e.g. executables, operating system elements, data bases. There are the system components with long life-cycle. Modifying any of these components forces ones to make considerable changes in the system (i.e., in its architecture, structure, and interfaces), to repetitive test and check the system security. Variable components are represented by occasionally modified system entities: e.g. security configuration settings (system registry values, access control rights, etc.), session elements (a list of running applications, etc.). There are the components with short life-cycle.

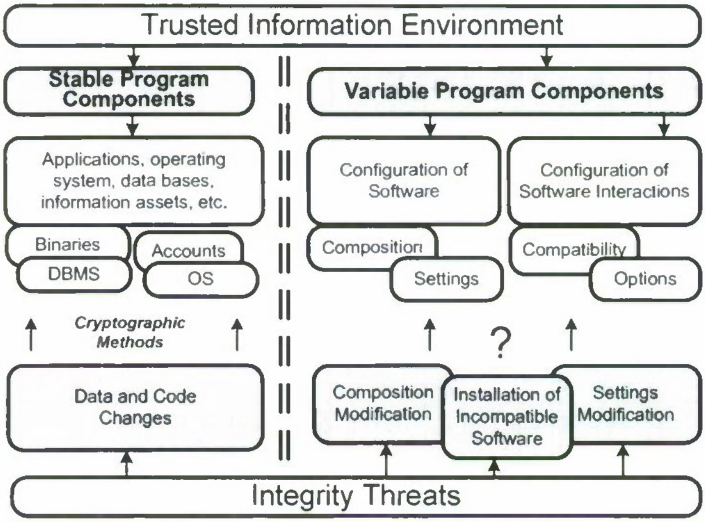

Trusted Information Environment
Stable Program Components
Variable Program Components
Applications, operating system, data bases, information assets, etc.
Binaries
Accounts
DBMS
OS
Configuration of Software
Configuration of Software Interactions
Composition
Settings
Compatibility
Options
Cryptographic Methods
?
Data and Code Changes
Composition Modification
Installation of Incompatible Software
Settings Modification
Integrity Threats

Fig. 1. TIE in Integrity Scope: Structure, Threats, and Control

Cryptographic integrity control methods are suitable only for the stable components. Variable ones do not undergo cryptographic approaches for regular changes provided in the system: for example, installation of incompatible software (e.g., a couple of cryptographic libraries which use different releases of system API); software update that induces the security re-configuration (e.g., Service Pack installation which results in access bits changed on folders and files); correction of the users list (e.g., adding a user leads to changes in the work environment, security settings, and user profiles). Therefore, if we want to reach trustiness for the system security we should control the system stability. As we can see, every change in the variable component is applied to the system security configuration and thus can be referred to as to a mutual agreement between the settings of TIE's components including system software, applications, and security mechanisms.

For better understanding the security and integrity aspects of the system trustiness, let's formally specify the solution of the integrity problem.

The system state is characterized with:

– a set of program components $p_i \in P$, where P depicts the set of TIE's components, $i \in N$. A program item is specified with a program type $T_n \in T$, where T is a set of program types (e.g., system software, user application, security mechanism), $n \in N$;

– a set of program attributes $A^{T_n} = \{a_j^{T_n}\}$, where T_n is a program type, a_j is a component of program attribute; $j \in N$. Program attributes are the settings of the TIE's program components;

– a set of attribute values $V^{T_n,p_i} = \{v_k^{T_n,p_i}\}$, where $\forall v_k^{T_n,p_i} = \mathrm{var}(p_i, T_n, A^{T_n})$, $k \in N$. Function $\mathrm{var}: P \times T \times A^T \to V^T$ for the program item $p_i \in P$ of type $T_n \in T$ with attributes A^{T_n} returns the values V^{T_n}.

To keep integrity, the system should, firstly, meet the security conditions at any of its security states and, secondly, the conditions of mutual agreement between the settings of TIE's components at any secure state. In other words, it means that every system state has to be secure and the security settings have to be agreed (compatible) with each other.

The security conditions are met in the system when it provides the security according to the security regulations (e.g., according to restrictions of security policy). Formally, the security control can be represented in the following manner (it is similar to discretionary and mandatory security models, but it is based on the predicative restrictions checking). The system Σ={ S^Σ ,*tr*, s_{init}^Σ ,Q} is a state machine, where S^Σ is a set of system states, $S^\Sigma = P \times T \times A^T \times V^T$; Q depicts a set of access queries; *tr* is a state transition function, $tr: S^\Sigma \to S^\Sigma$, which for the given access query $q \in Q$ transfers the system from the state s_x^Σ into the next state $s_{x+1}^\Sigma = tr(q, s_x^\Sigma)$; s_{init}^Σ is the initial state. The state s^Σ is called reachable in the system Σ iff there is a sequence <(q_0, s_0^Σ), ..., (q_n, s_n^Σ)>, where $s_0^\Sigma = s_{init}^\Sigma$, $s_n^\Sigma = s^\Sigma$, and $s_{x+1}^\Sigma = tr(q, s_x^\Sigma)$, $0 \le x < n$. For any system the state s_{init}^Σ is trivially reachable. In the most common case, the access control model M implemented in the system Σ can be represented as a set M = {S, R}, where S is a set of the model states (so called the security states), R is a set of access control rules. The access rules have a form of the predicates: $r(q,s,s')$. We define a function $pr: S^\Sigma \to S$ that specifies a correspondence between the system state and the security state. Predicate r checks that the result of the query q is a transfer of the system from the state s to the state s', i.e. there is the function $s^{\Sigma\prime} = tr(q, s^\Sigma), s' = pr(s^{\Sigma\prime}), s = pr(s^\Sigma), s^{\Sigma\prime} = pr^{-1}(s'), s^\Sigma = pr^{-1}(s)$ permitted by access control. Other words, transition of the system from the state $s^\Sigma \in S^\Sigma$ to the next state

$s^{\Sigma}{}' \in S^{\Sigma}$ is granted iff all predicates $r(q,s,s')$ which permit that transfer are true: $\forall s^{\Sigma}, s^{\Sigma}{}' \in S^{\Sigma} \quad \exists s, s' \in S : s = pr(s^{\Sigma}), s' = pr(s^{\Sigma}{}'), \forall r \in R : r(q,s,s') = \text{"TRUE"}$.

Property of security for the system Σ can be represented as $\Lambda = \{\Sigma, M, Cr\}$, where Cr is a set of security requirements (i.e. security criteria). The security constraints have a form of predicates like $cr(s)$ defined on the states S. These predicates check the security of the states. The state $s \in S$ is secure iff for each criterion $c \in C$ all of the predicates $c(s)$ are true: $\forall s^{\Sigma} \in S^{\Sigma} \quad \exists s \in S : s = pr(s^{\Sigma}), \forall c \in C : c(s) = \text{"TRUE"}$.

Therefore, formally, the system Σ which implements the access control model M meets the security conditions iff:

– the system Σ corresponds to the access control rules of the model M:

$$\forall s^{\Sigma}, s^{\Sigma}{}' \in S^{\Sigma} \quad \exists s, s' \in S : s = pr(s^{\Sigma}), s' = pr(s^{\Sigma}{}'), \exists r \in R : r(q,s,s') = \text{"TRUE"}$$

– the system states (i.e. a set of security settings and their values in the given state and in any reachable state) satisfy the security criteria:

$$\forall s^{\Sigma} \in S^{\Sigma} \quad \exists s \in S : s = pr(s^{\Sigma}), \forall c \in C : c(s) = \text{"TRUE"}.$$

Both of these issues can evident on system security.

To represent the system integrity via the security settings agreement between the program components, let's to review the function $ref : P \times T \times A^T \times V^T \to P \times T \times A^T \times V^T$ which for the set $a^t \in A^T$ with values $v^{t.p} \in V^{t.p}$ of the given program component $p \in P$ of the type $t \in T$ points to the set of agreed attributes $a^{t'} \in A^T$ with values $v^{t'.p'} \in V^{t'.p'}$ of another program item $p' \in P$ of the type $t' \in T$. In common case, for this function there is no restrictions like $p \neq p'$, because in the complex systems there is a mutual influence of the settings within sole program component (e.g., in the operating systems setting the values of some settings can refuse the action of other settings: for instance, in Windows, the registry key modification can suppress the Internet Explorer security option). Commonly, to each value of program item there is defined one (lets note it V^T) or several (lets depict it $V^T \pm \Delta V^T_{\Leftarrow}$) values referring to another program item: $ref : P \times T \times A \times V^T \to P \times T \times A \times (V^T \pm \Delta V^T_{\Leftarrow})$. The reverse function $ref^{-1} : P \times T \times A \times (V^T \pm \Delta V^T_{\Leftarrow}) \to P \times T \times A \times (V^T \pm \Delta V^T_{\Rightarrow})$ defines the area $V^T \pm \Delta V^T_{\Rightarrow}$ for each point taken from $V^T \pm \Delta V^T_{\Leftarrow}$. Existence of two areas $V^T \pm \Delta V^T_{\Leftarrow}$ and $V^T \pm \Delta V^T_{\Rightarrow}$ allows us formally specify the system integrity via agreement of the program items and their settings. Therefore, for the system that consists of a set of program items P, an area of symmetric relations has not to be empty:

$$\forall p \in P, \forall t \in T \quad \exists a \in A^t : \exists p' \in P, \exists t' \in T, \exists d_{\Rightarrow} = V^{p.T} \pm \Delta V^{p.T}_{\Rightarrow} \; u$$
$$d_{\Leftarrow} = V^{p.T} \pm \Delta V^{p.T}_{\Leftarrow} : ref(p,t,a^t,d_{\Rightarrow}) = < p',t',a^{t'},d_{\Leftarrow} >;$$
$$ref^{-1}(p',t',a^{t'},d_{\Leftarrow}) = < p,t,a^t,d' >; d' \cap d_{\Rightarrow} \neq \varnothing.$$

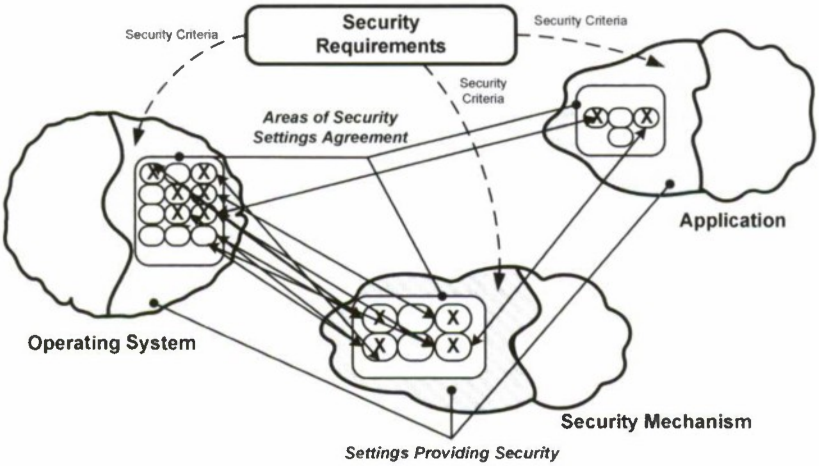


Fig. 2. Integrity Universe Finding

Intersection of all such areas provides the universe of integrity. Measure of integrity is a power of that universe. Fig. 2 demonstrates iterations of finding the integrity universe for three program components (operating system, application, and security mechanism).

If the process starts from one setting, then the area of agreement consists of one element, and as the result the TIE obtains the stable integrity. If there is an area of reverse settings, which intersection with the start area is not empty, then there are two possible variants:

– the reverse area re-calculation forms the parameters which are *not secure*. In that case, the system can not be considered as secure, because its configuration contains incompatible settings. As well, there is no integrity in that system;

– the reverse area re-calculation forms the parameters which *satisfy the security criteria*. In that case, there is necessary to recursively check all other program components for their agreement between the settings.

The discussed formalization makes it possible to compose the tools of dynamic security and integrity control for any kind of TIE (see Fig 3 for the common schema of the security and integrity control system). Historically, theoretical approaches aimed to build a system that allows to manage any process are summarized in the form of control systems. The control systems that provide automatic mode of maintenance are called automatic control systems (ACS) (e.g. [8], [9]). ACS is used to synthesize and analyze common models and specifications of mathematical and technical processes and systems. They do not touch problems of the information systems, especially the security aspects. We suggest combining theory of ACS and a concept of controllable settings and thus constructing an automatic security management system which monitors and controls the system security and integrity permanently in accordance to security requirements (e.g., in [10]) .

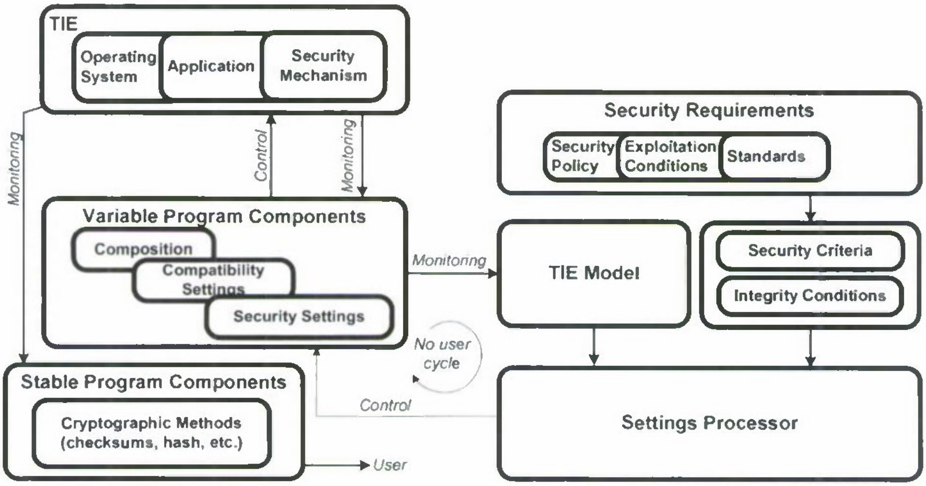


Fig. 3. The Dynamic Security and Integrity Control System

That system takes from ACS theory a paradigm of parameterized control applied to the target system. It is a closed-loop control system that requires no operator's action while it's working. This assumes the security estimation remains in the normal range for the controlled system. In our case, parameters to be controlled are the system security settings and their changeable values: a set and a structure of the critical system program components (i.e. applications, services, executive files, processes, etc); a set and a structure of the access subjects (i.e. users, groups, members of groups, etc.); a set and a structure of the critical access objects (i.e. files, directories, registry keys, printers, shared resources, hierarchical structure, etc.); a set and values of the subjects' and objects' security settings (i.e. names, paths, IDs, privileges, access rights, owners IDs, etc.); a set and values and security options of the applications (i.e. Internet security zones, login/passwords, firewall filtering rules, etc.)

A set of security parameters is called a system configuration. The system configuration is a manipulated variable (terminology of ACS). Another variable — a controlled variable — is a security and integrity estimation. It is maintained at a specified value or within a specified range. To control security, the system acts on the configuration to maintain the security and integrity estimation at the specified value or within the specified range. The control system gets information of the current system security configuration; evaluates the security of the current configuration; estimates the integrity of the current configuration over all program components; and adapts the system's configuration to security impacts. In that manner we obtain automatic implementation of the permanent active cycle of security management applied to the practical IE and thus make IE a trusted one.

4 Conclusion

The paper has addressed to security problem of integrity monitoring and control in modern complex information environments. We have reviewed that the environment contains variable and stable components. As for stable items, there are no innovations; the cryptography algorithms are well suitable for integrity control. But for the variable components the new approaches are required.

The paper has discussed the formal foundations of the suggested method of integrity control for changeable program components of the trusted information environment. Our technique is based on finding the security settings which form the secure environment, and on consequent iterative searching of secure settings which are mutually agreed with all settings of all program components. The suggested approach allowed us to propose a schema of dynamic Security and Integrity Control System which could automate process of security trustiness assurance.

References

1. ISO/IEC 15408: Information technology. Security techniques. Evaluation criteria for IT security (2005)
2. Trusted Solaris, `http://www.sun.com/software/solaris/trustedsolaris`
3. HPUX, `http://docs.hp.com`
4. RSBAC, `http://www.rsbac.org/documentation/rsbac_handbook`
5. GRSecurity, `http://grsecurity.org`
6. Astaro Security Gateway, `http://www.astaro.com/our_products/astaro_security_gateway`
7. Alves-Foss, J., Taylor, C., Oman, P.: A multi-layered approach to security in high assurance system development. In: Proc. of 37th Annual Hawaii Intl. Conf. on System Science (HICSS-37), Hawaii (2004)
8. Stefani, R.T., et al.: Design of Feedback Control Systems. Oxford University Press, Oxford (2001)
9. Dorf, R.C., Bishop, R.H.: Modern Control Systems. Prentice Hall, Englewood Cliffs (2007)
10. ISO/IEC 17799: Information technology. Security techniques. Code of practice for information security management (2005)
11. Dillard, K., Maldonado, J., Warrender, B.: Microsoft Solutions for Security. Windows Server 2003 Security Guide. Microsoft (2003)

Author Index

Lecture Notes in Computer Science

The LNCS series reports state-of-the-art results in computer science research, development, and education, at a high level and in both printed and electronic form. Enjoying tight cooperation with the R&D community, with numerous individuals, as well as with prestigious organizations and societies, LNCS has grown into the most comprehensive computer science research forum available.

The scope of LNCS, including its subseries LNAI and LNBI, spans the whole range of computer science and information technology including interdisciplinary topics in a variety of application fields. The type of material published traditionally includes

- proceedings (published in time for the respective conference)
- post-proceedings (consisting of thoroughly revised final full papers)
- research monographs (which may be based on outstanding PhD work, research projects, technical reports, etc.)

More recently, several color-cover sublines have been added featuring, beyond a collection of papers, various added-value components; these sublines include

- tutorials (textbook-like monographs or collections of lectures given at advanced courses)
- state-of-the-art surveys (offering complete and mediated coverage of a topic)
- hot topics (introducing emergent topics to the broader community)

In parallel to the printed book, each new volume is published electronically in LNCS Online.

Detailed information on LNCS can be found at
www.springer.com/lncs

Proposals for publication should be sent to
LNCS Editorial, Tiergartenstr. 17, 69121 Heidelberg, Germany
E-mail: lncs@springer.com

ISSN 0302-9743

ISBN 978-3-642-14705-0

9 783642 147050

Lecture Notes in Computer Science

LNCS LNAI LNBI

› springer.com